Efficiently deciding equivalence for standard primitives and phases

Véronique Cortier, Antoine Dallon, Stéphanie Delaune

Efficiently deciding equivalence for standard primitives and phases

Véronique Cortier[1], Antoine Dallon[1,2,3], and Stéphanie Delaune[3]

[1] LORIA, CNRS, France
[2] LSV, CNRS & ENS Paris-Saclay, France
[3] Univ Rennes, CNRS, IRISA, France

Abstract. Privacy properties like anonymity or untraceability are now well identified, desirable goals of many security protocols. Such properties are typically stated as equivalence properties. However, automatically checking equivalence of protocols often yields efficiency issues.
We propose an efficient algorithm, based on graph planning and SAT-solving. It can decide equivalence for a bounded number of sessions, for protocols with standard cryptographic primitives and phases (often necessary to specify privacy properties), provided protocols are well-typed, that is encrypted messages cannot be confused. The resulting implementation, SAT-Equiv, demonstrates a significant speed-up w.r.t. other existing tools that decide equivalence, covering typically more than 100 sessions. Combined with a previous result, SAT-Equiv can now be used to prove security, for some protocols, for an unbounded number of sessions.

1 Introduction

Security protocols are notoriously difficult to design. A common good practice is to formally analyse protocols using symbolic techniques, in order to spot flaws possibly before their deployment (e.g. TLS 1.3 [23, 5], an avionic protocol [7]). These symbolic techniques are mature for reachability properties like confidentiality or authentication. More recently, this approach has been extended to privacy properties, such as vote secrecy, anonymity, untraceability, or unlinkability. These properties are expressed through equivalences. For example, in the case of biometric passports, an attacker should not be able to distinguish whether she is in contact with Alice's passport or Bob's passport.

Recently, a new tool, SAT-Equiv [20], has been proposed to decide such equivalence properties for security protocols, for a bounded number of sessions. It is based on a standard model-checking approach, namely graph planning [9, 27] and SAT-solving. Intuitively, protocols executions are over-approximated as a graph planning problem, which allows to consider several possible interleavings in parallel, allowing the analysis of dozen of sessions of a protocol in a few seconds. However, this result is limited to a very small set of primitives, namely symmetric encryption and concatenation.

Our contributions. Building upon this novel approach, we enrich SAT-Equiv in order to cover protocols using asymmetric primitives and/or phases. As for the original SAT-Equiv, we assume a non confusion property: encrypted messages should not be confused, a condition automatically checked by our tool and which can be enforced e.g. through appropriate labelling.

First, we extend SAT-Equiv to cover all standard primitives: symmetric and asymmetric encryption, signatures, and hashes. Since graph planning is a bounded model-checking technique, SAT-Equiv relies on a small model property, that bounds the size of messages. More precisely, [14] guarantees that if there is an attack, then there is a well-typed attack, where messages follow a fix format. This result has been recently extended to standard primitives [17]. The straightforward extension of SAT-Equiv to standard primitives however yields severe efficiency issues. Indeed, unlike the symmetric encryption case, checking whether two sequences of messages are equivalent (i.e. in static equivalence) may require complex tests where the attacker *construct* messages (that is, hash or asymmetrically encrypt messages). We therefore provide a precise characterisation of the set of tests that need to be considered when checking for static equivalence. This characterisation is of independent interest and could be used in other contexts. We also extend SAT-Equiv to consider protocols with phases, which are useful to model game-based properties.

Our extension of SAT-Equiv now provably terminates. In [20], termination can be guaranteed by checking that any state of the planning graph is indeed reachable, which requires to query a SAT-solver at each step. While this provides termination in theory, this yields a non practical algorithm and has not been implemented. Instead, we exhibit a bound on the maximal length of the smallest attack (bounding the attacker steps as well). It is therefore sufficient to stop the construction of the graph planning once this bound has been reached, enforcing termination for free (no computation overhead).

Finally, we have considerably revisited and improved the original implementation of SAT-Equiv. This significant speedup now allows for security proofs for an unbounded number of sessions. Indeed, [16] shows decidability of equivalence, for an unbounded number of sessions, for protocols with an acyclic dependency graph. The notion of dependency graph is introduced in [16] and intuitively captures how the input/output actions of the protocol may use messages from other steps of the protocol. As a corollary, [16] induces a bound on the number of sessions that needs to be considered for an attack, which depends on the size and structure of the graph. This bound can be rather large (50 to 100 sessions, even on small examples) but SAT-Equiv is now able to reach such bounds.

These novelties are implemented in an extension of SAT-Equiv and compared with the other tools of the literature, namely Spec [24], Akiss [10] and the very recent DeepSec [13] tool. Our experiments show that SAT-Equiv is much faster on all the examples, allowing to reach typically more than 100 sessions. As an application, we consider two protocols, Denning-Sacco and Needham-Schroeder symmetric keys, shown to have acyclic dependency graphs in [16]. Considering

the necessary number of sessions as induced by [16], we establish trace equivalence for these two protocols, for an unbounded number of sessions.

The tool source, the example files as well as all the proofs are available in [1].

Related work. There are two main families of tools to analyse equivalence properties on security protocols. Some tools prove equivalence for an arbitrary number of sessions, that is, no matter how often a protocol is used. The main tools in this category are ProVerif [8], Tamarin [28], Maude-NPA [26], Type-Eq [21]. Maude-NPA often suffers from termination issues when used for equivalence properties. Type-Eq [21, 22] is a sound (but incomplete) type-checker for equivalence properties that has good performance. It requires that protocols have a similar structure. ProVerif and Tamarin work well in practice. They actually prove a stronger notion of equivalence, diff-equivalence, that also requires that the two considered protocols have a very similar structure. Moreover, equivalence properties are undecidablein general for an unbounded number of sessions. Therefore, ProVerif may not terminate and Tamarin may need some user guidance.

A second approach consists in *deciding* equivalence, for a bounded number of sessions. Spec [24] is one of the first tool that decides equivalence of security protocols but it does not scale well when the number of sessions grows (it can typically handle up to three sessions for small protocols). DeepSec [13] is a very recent tool that builds upon Akiss [10] and Apte [11]. All these tools analyse symbolic executions and typically have to consider all possible interleavings between the roles of the protocol, which often raises efficiency issues.

2 Model

Protocols are modeled through a process algebra, in the spirit of the applied-pi calculus [2]. We consider here the model used in [20, 17].

2.1 Term algebra

As usual, messages are modeled by terms. Private data are represented through an infinite set $\mathcal{N}$ of *names* used to model *e.g.* keys or nonces. We consider an infinite set $\mathcal{C}_0$ of constants to represent public data such as agent names or attacker's nonces or keys. We consider also two sets of *variables* $\mathcal{X}$ and $\mathcal{W}$. Variables in $\mathcal{X}$ model arbitrary data expected by the protocol, while variables in $\mathcal{W}$ are used to store messages learnt by the attacker. A *data* is either a constant, a variable, or a name. Cryptograhic primitives are represented by function symbols. We consider the *signature* Σ parameterised by $n \geq 2$:

- $\Sigma_{\mathsf{c}} = \{\mathsf{senc},\ \mathsf{aenc},\ \mathsf{hash},\ \mathsf{pub},\ \mathsf{sign},\ \mathsf{vk},\ \mathsf{ok}\} \cup \{\langle\ \rangle_k \mid 2 \leq k \leq n\}$;
- $\Sigma_{\mathsf{d}} = \{\mathsf{sdec},\ \mathsf{adec},\ \mathsf{getmsg}\} \cup \{\mathsf{proj}_j^k \mid 2 \leq k \leq n \text{ and } 1 \leq j \leq k\}$; and
- $\Sigma = \Sigma_{\mathsf{c}} \cup \Sigma_{\mathsf{d}} \cup \{\mathsf{check}\}$.

The symbols senc, aenc, sdec, and adec of arity 2 are used to model resp. symmetric and asymmetric encryption. We also consider signature sign and hash

function hash. Concatenation of messages is modeled through tuple operators together with their projection functions. For example, $\langle m_1, m_2, m_3\rangle_3$ represents the concatenation of the three messages m_1, m_2, and m_3. It is syntactically different from the nested pairs $\langle m_1, \langle m_2, m_3\rangle_2\rangle_2$. These two representations correspond to different implementation choices. We distinguish between *constructors* in Σ_{c} and *destructors* in Σ_{d}. The symbol check of arity 2, that corresponds to the verification of a signature, is neither a destructor nor a constructor. The set of terms built from a signature $\mathcal{F}$ and a set of data D is denoted $\mathcal{T}(\Sigma, D)$. Given a term u, we denote $St(u)$ the set of its *subterms*, $vars(u)$ the set of its variables, and $\mathsf{root}(u)$ its root symbol. A term is *ground* if it contains no variable. The application of a substitution σ to a term u is written $u\sigma$. We denote $dom(\sigma)$ its *domain* and $img(\sigma)$ its *image*. Two terms u_1 and u_2 are *unifiable* when there exists a substitution σ such that $u_1\sigma = u_2\sigma$.

We consider two *sorts*: atom and $\mathsf{bitstring}$. The sort atom represents atomic data like nonces or keys while $\mathsf{bitstring}$ models arbitrary messages. Names in $\mathcal{N}$ and constants in $\mathcal{C}_0$ have sort atom. Any $\mathsf{f} \in \Sigma_{\mathsf{c}}$ comes with its sorted arity:

$$\begin{array}{ll}
\langle\ \rangle_k : \mathsf{bitstring} \times \cdots \times \mathsf{bitstring} \to \mathsf{bitstring} & \mathsf{ok} : \quad \to \mathsf{bitstring} \\
\mathsf{senc} : \mathsf{bitstring} \times \mathsf{atom} \to \mathsf{bitstring} & \mathsf{pub} : \mathsf{atom} \to \mathsf{bitstring} \\
\mathsf{aenc} : \mathsf{bitstring} \times \mathsf{bitstring} \to \mathsf{bitstring} & \mathsf{vk} : \mathsf{atom} \to \mathsf{bitstring} \\
\mathsf{sign} : \mathsf{bitstring} \times \mathsf{atom} \to \mathsf{bitstring} & \mathsf{hash} : \mathsf{bitstring} \to \mathsf{bitstring}
\end{array}$$

Given $D \subseteq \mathcal{C}_0 \uplus \mathcal{X}$, the set $\mathcal{T}_0(\Sigma_{\mathsf{c}}, D)$ is the set of terms t in $\mathcal{T}(\Sigma_{\mathsf{c}}, D)$ such that *(i)* for any term $\mathsf{pub}(u)$ (resp. $\mathsf{vk}(u)$) in $St(t)$, u is of sort atom; *(ii)* for any $\mathsf{aenc}(u, v) \in St(t)$, $v = \mathsf{pub}(v')$ for some v'. Terms in $\mathcal{T}_0(\Sigma_{\mathsf{c}}, \mathcal{N} \uplus \mathcal{C}_0)$ are called *messages*. Intuitively, messages are terms with atomic keys.

The properties of the cryptographic primitives are reflected through the following convergent rewriting rules.

$$\begin{array}{ll}
\mathsf{sdec}(\mathsf{senc}(x, y), y) \to x & \mathsf{adec}(\mathsf{aenc}(x, \mathsf{pub}(y)), y) \to x \\
\mathsf{getmsg}(\mathsf{sign}(x, y)) \to x & \mathsf{check}(\mathsf{sign}(x, y), \mathsf{vk}(y)) \to \mathsf{ok} \\
\mathsf{proj}_j^k(\langle x_1, \ldots, x_k\rangle_k) \to x_j & \text{with } 2 \leq k \leq n \text{ and } 1 \leq j \leq k
\end{array}$$

A term u can be rewritten into v if there is a position p in u, and a rewriting rule $\mathsf{g}(t_1, ..., t_n) \to t$ such that $u|_p = \mathsf{g}(t_1, \ldots, t_n)\theta$ for some substitution θ, and $v = u[t\theta]_p$, i.e. u in which the subterm at position p has been replaced by $t\theta$. Moreover, we assume that $t_1\theta, \ldots, t_n\theta$ as well as $t\theta$ are messages, in particular they do not contain destructor symbols. As usual, we denote $\to^*$ the reflexive-transitive closure of $\to$, and $u{\downarrow}$ the normal form of a term u.

An attacker builds his own messages by applying public function symbols to terms he already knows and that are available through variables in $\mathcal{W}$. Formally, a computation done by the attacker is a *recipe*, *i.e.* a term in $\mathcal{T}(\Sigma, \mathcal{W} \uplus \mathcal{C}_0)$.

2.2 Process algebra

We consider processes that may receive and send messages. We assume that each process communicates on a dedicated public channel. In practice, ip adresses and sessions identifiers are typically used to desambiguate which message is adressed

to who and for which session. Of course, these channels may be freely manipulated by the attacker. Since we consider equivalence properties, distinct (public) channels provide more abilities for the adversary to distinguish between protocols. Formally, given a set $\mathcal{C}h$ of channels, we consider the fragment of simple processes without replication built on basic processes as defined *e.g.* in [12].

Definition 1. *A* basic processes *is defined as follows:*

$$P, Q := 0 \mid \mathsf{in}(c, u_1).P \mid \mathsf{out}(c, u_2).P \mid i{:}P$$

with $u_1, u_2 \in \mathcal{T}_0(\Sigma_c, \mathcal{C}_0 \uplus \mathcal{N} \uplus \mathcal{X})$, $c \in \mathcal{C}h$, *and increasing phase numbers. A* simple process *is a multiset of basic processes on pairwise distinct channels. A* protocol *is a simple process such that all its variables are in the scope of an input.*

The process 0 does nothing and we often omit it. The process "$\mathsf{in}(c, u_1).P$" expects a message m of the form u_1 on channel c and then behaves like $P\sigma$ where σ is a substitution such that $m = u_1\sigma$. Note that checking whether a received message has the expected form is done through pattern-matching instead of explicit tests. The process "$\mathsf{out}(c, u_2).P$" emits u_2 on c, and then behaves like P. Our calculus also has a phase instruction, in the spirit of [8], denoted $i{:}P$. This instruction is useful to model security requirements, for example in case the attacker interacts with the protocol before being given some secret.

Example 1. As an illustrative example, we consider a simplified version of the Denning-Sacco protocol which is a key distribution protocol relying on asymmetric encryption and signature. Informally, the protocol is as follows.

$$A \to B : \mathsf{aenc}(\mathsf{sign}(\langle A, B, K_{ab}\rangle, \mathsf{prv}(A)), \mathsf{pub}(B))$$

The agents A and B aim at authenticating each other and establishing a fresh session key K_{ab}. We model this protocol in our formalism through the simple process $\mathcal{P}_{\mathsf{DS}} = \{P_A; P_B\}$ where $P_A = \mathsf{out}(c_A, \mathsf{aenc}(\mathsf{sign}(\langle \mathsf{a}, \mathsf{b}, k_{ab}\rangle_3, sk_a), \mathsf{pub}(sk_b))).0$ and $P_B = \mathsf{in}(c_B, \mathsf{aenc}(\mathsf{sign}(\langle \mathsf{a}, \mathsf{b}, x\rangle_3, sk_a), \mathsf{pub}(sk_b))).0$ where sk_a, sk_b, and k_{ab} are names, a and b are constants, and x is a variable.

The operational semantics of a process is defined using a relation over configurations. A *configuration* is a tuple $(\mathcal{P}; \phi; \sigma; i)$ with $i \in \mathbb{N}$ and such that:

- $\mathcal{P}$ is a multiset of processes (not necessarily ground);
- $\phi = \{\mathsf{w}_1 \triangleright m_1, \ldots, \mathsf{w}_n \triangleright m_n\}$ is a *frame*, *i.e.* a substitution where $\mathsf{w}_1, \ldots, \mathsf{w}_n$ are variables in $\mathcal{W}$, and $m_1, \ldots, m_n$ are messages;
- σ is a substitution such that $dom(\sigma) = fv(\mathcal{P})$, and $img(\sigma)$ are messages.

Intuitively, $\mathcal{P}$ represents the processes that still remain to be executed; ϕ represents the sequence of messages that have been learnt so far by the attacker, and σ stores the value of the variables that have already been instantiated. We write P instead of $0{:}P$ and $P \uplus \mathcal{P}$ instead of $\{P\} \uplus \mathcal{P}$. Given a protocol $\mathcal{P}$, we also often write $\mathcal{P}$ instead of $(\mathcal{P}; \emptyset; \emptyset; 0)$. The operational semantics is induced by the relation $\xrightarrow{\alpha}$ over configurations defined in Figure 1. For example, the In rule defines how messages can be input on a (public) channel: the adversary may

$$\textsc{In} \quad (i{:}\mathsf{in}(c,u).P \cup \mathcal{P};\phi;\sigma;i) \xrightarrow{\mathsf{in}(c,R)} (i{:}P \cup \mathcal{P};\phi;\sigma \uplus \sigma_0;i)$$
where R is a recipe such that $R\phi{\downarrow}$ is a message, and $R\phi{\downarrow} = (u\sigma)\sigma_0$ for σ_0 with $dom(\sigma_0) = vars(u\sigma)$.

$$\textsc{Out} \quad (i{:}\mathsf{out}(c,u).P \cup \mathcal{P};\phi;\sigma;i) \xrightarrow{\mathsf{out}(c,\mathsf{w})} (i{:}P \cup \mathcal{P};\phi \cup \{\mathsf{w} \triangleright u\sigma\};\sigma;i)$$
with w a fresh variable from $\mathcal{W}$, and $u\sigma$ is a message.

$$\textsc{Move} \quad (\mathcal{P};\phi;\sigma;i) \xrightarrow{\mathsf{phase}\ i'} (\mathcal{P};\phi;\sigma;i') \qquad \text{with } i' > i.$$

$$\textsc{Phase} \quad (i'{:}i''{:}P \cup \mathcal{P};\phi;\sigma;i) \xrightarrow{\tau} (i''{:}P \cup \mathcal{P};\phi;\sigma;i)$$

Fig. 1. Semantics for processes

send any message, provided she can construct it through a recipe R applied on her previous knowledge ϕ. Note that only messages can be received (and sent). The relation $\xrightarrow{\mathsf{tr}}$ between configurations (where tr is a possibly empty sequence of actions) is defined in the usual way. Given a configuration $\mathcal{K}$, we write:

$$\mathsf{trace}(\mathcal{K}) = \{(\mathsf{tr},\phi) \mid \mathcal{K} \xrightarrow{\mathsf{tr}} (\mathcal{P}';\phi;\sigma;i) \text{ for some configuration } (\mathcal{P}';\phi;\sigma;i)\}.$$

Example 2. Continuing Example 1, let $\mathcal{K}_{\mathsf{DS}} = (\{P_A; P_B; P_{B'}\};\phi_0;\emptyset;0)$ where $P_{B'}$ models an additional session of the role B obtained by simply renaming c_B and x with c'_B and x'. The frame $\phi_0 = \{\mathsf{w}_a \triangleright \mathsf{vk}(sk_a), \mathsf{w}_b \triangleright \mathsf{pub}(sk_b)\}$ models the fact that the attacker initially knows the public key of b and the verification key of a. We consider a simple scenario without dishonest participant. The trace $\mathsf{tr}_0 = \mathsf{out}(c_A,\mathsf{w}_1).\mathsf{in}(c_B,\mathsf{w}_1).\mathsf{in}(c'_B,\mathsf{w}_1)$ is executable from $\mathcal{K}_{\mathsf{DS}}$, and yields $\phi = \phi_0 \uplus \{\mathsf{w}_1 \triangleright \mathsf{aenc}(\mathsf{sign}(\langle \mathsf{a},\mathsf{b},k_{ab}\rangle_3, sk_a), \mathsf{pub}(sk_b))\}$, i.e. $(\mathsf{tr}_0,\phi) \in \mathsf{trace}(\mathcal{K}_{\mathsf{DS}})$.

2.3 Type-compliance

We present here our main assumption on protocols. Intuitively, we assume that ciphertexts cannot be confused, and we rely for this on a notion of typing system.

Definition 2. *A* typing system *is a pair $(\mathcal{T}_{\mathsf{init}},\delta)$ where $\mathcal{T}_{\mathsf{init}}$ is a set of elements called* initial types*, and δ is a function mapping data in $\mathcal{C}_0 \uplus \mathcal{N} \uplus \mathcal{X}$ to types τ:*

$$\tau,\tau_1,\tau_2 = \tau_0 \mid \mathsf{f}(\tau_1,\ldots,\tau_n) \text{ with } \mathsf{f} \in \Sigma_{\mathsf{c}} \text{ and } \tau_0 \in \mathcal{T}_{\mathsf{init}}$$

Then, δ is extended to constructor terms as follows:

$$\delta(\mathsf{f}(t_1,\ldots,t_n)) = \mathsf{f}(\delta(t_1),\ldots,\delta(t_n)) \text{ with } \mathsf{f} \in \Sigma_{\mathsf{c}}.$$

A configuration is type-compliant if two unifiable encrypted subterms have the same type. We write $ESt(t)$ for the set of *encrypted subterms* of t, i.e. $ESt(t) = \{u \in St(t) \mid u \text{ is of the form } \mathsf{f}(u_1,\ldots,u_n) \text{ and } \mathsf{f} \neq \langle\ \rangle_i\}$.

Definition 3. *A configuration $\mathcal{K}$ is type-compliant w.r.t. a typing system $(\mathcal{T}_{\mathsf{init}},\delta)$ if for every $t,t' \in ESt(\mathcal{K})$ we have that t and t' unifiable implies that $\delta(t) = \delta(t')$.*

Example 3. Continuing our running example, we consider the typing system generated from $\mathcal{T}_{\mathsf{DS}} = \{\tau_{\mathsf{a}},\tau_{\mathsf{b}},\tau_k,\tau_{sk}\}$ of initial types, and the function δ_{DS} that associates the expected type to each constant/name ($\delta_{\mathsf{DS}}(\mathsf{a}) = \tau_{\mathsf{a}}$, $\delta_{\mathsf{DS}}(k_{ab}) = \tau_k$,

etc), and such that $\delta_{\mathsf{DS}}(x) = \delta_{\mathsf{DS}}(x') = \tau_k$. We have that $\mathcal{K}_{\mathsf{DS}}$ is type-compliant w.r.t. $(\mathcal{T}_{\mathsf{DS}}, \delta_{\mathsf{DS}})$: unifiable encrypted subterms occurring in the configuration have the same type since $\delta_{\mathsf{DS}}(x) = \delta_{\mathsf{DS}}(x') = \delta_{\mathsf{DS}}(k_{ab})$.

Type-compliant protocols have the property that, when looking for attacks, it is sufficient to consider well-typed execution. Formally, an execution $\mathcal{K} \xrightarrow{\mathsf{tr}} (\mathcal{P}; \phi; \sigma; i)$ is well-typed w.r.t. a typing system $(\mathcal{T}_{\mathsf{init}}, \delta)$, if σ is a well-typed substitution, i.e. every variable of its domain has the same type as its image.

2.4 Trace equivalence

Many privacy properties such as vote-privacy or untraceability are expressed as trace equivalence [25, 3]. Intuitively, two configurations are trace equivalent if an attacker cannot tell with which of the two configurations he is interacting. We first introduce a notion of equivalence (actually, inclusion) between frames.

Definition 4. *Two frames ϕ_1 and ϕ_2 are in* static inclusion*, written $\phi_1 \sqsubseteq_s \phi_2$, when $dom(\phi_1) = dom(\phi_2)$, and:*
- *for any recipe R, we have that $R\phi_1{\downarrow}$ is message implies that $R\phi_2{\downarrow}$ is message;*
- *for any recipes R, R' such that $R\phi_1{\downarrow}$, $R'\phi_1{\downarrow}$ are messages, we have that: $R\phi_1{\downarrow} = R'\phi_1{\downarrow}$ implies $R\phi_2{\downarrow} = R'\phi_2{\downarrow}$.*

Intuitively, ϕ_1 is included in ϕ_2 if any recipe producing a message in ϕ_1 also produces a message in ϕ_2 and if any equality satisfied in ϕ_1 is also satisfied in ϕ_2.

Example 4. We consider $\phi_1 = \phi \uplus \{\mathsf{w}_2 \triangleright \mathsf{senc}(\mathsf{m}_1, k_{ab}), \mathsf{w}'_2 \triangleright \mathsf{senc}(\mathsf{m}_1, k_{ab})\}$, and $\phi_2 = \phi \uplus \{\mathsf{w}_2 \triangleright \mathsf{senc}(\mathsf{m}_2, k), \mathsf{w}'_2 \triangleright \mathsf{senc}(\mathsf{m}_2, k')\}$ where $\mathsf{m}_1, \mathsf{m}_2 \in \mathcal{C}_0$. We have that $\mathsf{w}_2\phi_1{\downarrow} = \mathsf{w}'_2\phi_1{\downarrow}$ whereas this equality does not hold in ϕ_2. Hence $\phi_1 \not\sqsubseteq_s \phi_2$.

Trace inclusion is the active counterpart of static inclusion. Two configurations are in trace inclusion if, however the attacker behaves, the resulting sequences of messages observed by the attacker are in static inclusion.

Definition 5. *Let $\mathcal{K}$ and $\mathcal{K}'$ be two configurations. We have that $\mathcal{K} \sqsubseteq_t \mathcal{K}'$, if for every $(\mathsf{tr}, \phi) \in \mathsf{trace}(\mathcal{K})$, there exists $(\mathsf{tr}, \phi') \in \mathsf{trace}(\mathcal{K}')$ such that $\phi \sqsubseteq_s \phi'$.*

We easily derive a notion of trace equivalence: two configurations $\mathcal{K}$ and $\mathcal{K}'$ are trace equivalence, denoted $\mathcal{K} \approx_t \mathcal{K}'$, if $\mathcal{K} \sqsubseteq_t \mathcal{K}'$ and $\mathcal{K}' \sqsubseteq_t \mathcal{K}$. This notion of trace equivalence slightly differs from the one used in e.g. [14] but they actually coincide on the class of protocols we consider in this paper [10].

Example 5. To model the secrecy of the key k_{ab}, we define *strong secrecy* of k_{ab} by requiring that k_{ab} is indistinguishable from a fresh value. Formally, we consider P^1_B (resp. $P^1_{B'}$) obtained by replacing the process 0 with $1{:}\mathsf{out}(c_B, \mathsf{senc}(m_1, x))$ (resp. $1{:}\mathsf{out}(c'_B, \mathsf{senc}(m_1, x'))$). On the other side of the equivalence, we consider P^2_B and $P^2_{B'}$ obtained by replacing the process 0 with $1{:}\mathsf{out}(c_B, \mathsf{senc}(m_2, k))$ (resp. $1{:}\mathsf{out}(c'_B, \mathsf{senc}(m_2, k'))$) with fresh names k and k'.

$$\mathcal{K}^1_{\mathsf{DS}} = (\{P_A; P^1_B; P^1_{B'}\}; \phi_0) \text{ and } \mathcal{K}^2_{\mathsf{DS}} = (\{P_A; P^2_B; P^2_{B'}\}; \phi_0).$$

Then, we can show that $\mathcal{K}^1_{\mathsf{DS}} \not\sqsubseteq \mathcal{K}^2_{\mathsf{DS}}$ since k_{ab} is not strongly secret. An attacker can replay the message sent by A due to lack of freshness. This is exemplified by the trace $\mathsf{tr}_0.\mathsf{out}(c_B, \mathsf{w}_2).\mathsf{out}(c'_B, \mathsf{w}'_2)$ and the test given in Example 4.

3 From static inclusion to planning

The overall objective of this paper is to provide a practical algorithm for deciding trace inclusion (and thus trace equivalence) relying on graph planning and SAT solving. We start here by explaining how to build a planning problem to two frames such that the planning problem has a solution if, and only if, the two corresponding frames are not in static inclusion.

3.1 Planning problems

We first recall the definition of a planning problem, slightly simplified from [18]. Intuitively, a planning system defines a transition system from sets of facts to sets of facts. New facts may be produced and some old facts may be deleted.

Definition 6. *A* planning system *is tuple* $\langle \mathcal{F}act, \mathcal{I}nit, \mathcal{R}ule \rangle$ *where* $\mathcal{F}act$ *is a set of ground formulas called* facts*,* $\mathcal{I}nit_0 \subseteq \mathcal{F}act$ *is a set of facts representing the initial state, and* $\mathcal{R}ule$ *is a set of rules of the form* $\mathsf{Pre} \to \mathsf{Add}; \mathsf{Del}$ *where* Pre*,* Add*,* Del *are finite sets of facts such that* $\mathsf{Add} \cap \mathsf{Del} = \emptyset$*,* $\mathsf{Del} \subseteq \mathsf{Pre}$*. We write* $\mathsf{Pre} \to \mathsf{Add}$ *when* $\mathsf{Del} = \emptyset$*.*

Given a rule $r \in \mathcal{R}ule$ of the form $\mathsf{Pre} \to \mathsf{Add}; \mathsf{Del}$, we denote $\mathsf{Pre}(r) = \mathsf{Pre}$, $\mathsf{Add}(r) = \mathsf{Add}$, and $\mathsf{Del}(r) = \mathsf{Del}$. If $S \subseteq \mathcal{F}act$ are such that $\mathsf{Pre}(r) \subseteq S$, then we say that the rule is *applicable* in S, denoted $S \xrightarrow{r} S'$, and the state $S' = (S \setminus \mathsf{Del}) \cup \mathsf{Add}$ is the state resulting from the application of r to S. We allow some rules to be applied in parallel when no facts are deleted. Given $S \subseteq \mathcal{F}act$, and a set of rules $\{r_1, \ldots, r_k\}$ such that $\mathsf{Del}(r_i) = \emptyset$ and $\mathsf{Pre}(r_i) \subseteq S$ for any $i \in \{1, \ldots, k\}$, $\{r_1, \ldots, r_k\}$ is *applicable* in S, denoted $S \xrightarrow{\{r_1, \ldots, r_k\}} S'$, and the state $S' = \bigcup_{i=1}^{k} \mathsf{Add}(r_i) \cup S$ is the state resulting from the application of $\{r_1, \ldots, r_k\}$ to S.

A *planning path* from $S_0 \subseteq \mathcal{F}act$ to $S_n \subseteq \mathcal{F}act$ is a sequence $r_1, \ldots, r_n$ made of rules or sets of rules in $\mathcal{R}ule$ such that $S_0 \xrightarrow{r_1} S_1 \xrightarrow{r_2} \ldots S_{n-1} \xrightarrow{r_n} S_n$ for some stated $S_1, \ldots, S_{n-1} \subseteq \mathcal{F}act$. A *planning problem* for a system $\Theta = \langle \mathcal{F}act, \mathcal{I}nit, \mathcal{R}ule \rangle$ is a pair $\Pi = \langle \Theta, S_f \rangle$ where $S_f \subseteq \mathcal{F}$ represents the target facts. A solution to $\Pi = \langle \Theta, S_f \rangle$, called a *plan*, is a planning path from $\mathcal{I}nit$ to a state S_n such that $S_f \subseteq S_n$.

A transition $S \xrightarrow{\{r_1, \ldots, r_k\}} S'$ can be mimicked by $S \xrightarrow{r_1} S_1 \xrightarrow{r_2} \ldots \xrightarrow{r_k} S'$, thus the possibility of applying set of rules in a single step does not change the set of reachable states from a given state S. However, this allows us to consider plans of smaller length and will be useful later on to derive a tight bound and ensure the termination of our algorithm.

In this section, we explain the translation of static inclusion into a planning problem. We consider an (infinite) set $\mathcal{F}act_0$ of facts that represent the attacker's knowledge, i.e. formulas of the form $\mathsf{att}(u_P, u_Q)$ where u_P and u_Q are messages, plus a special symbol bad. Intuitively, $\mathsf{att}(u_P, u_Q)$ represents the fact that the attacker knows u_P in the "left" frame, while he knows u_Q in the "right" one.

3.2 Attacker analysis rules

Following [20], we first describe the planning rules that correspond to the analysis part of the attacker behaviours. We start by describing a set of abstract rules $\mathsf{R}_{\mathsf{Ana}}$ that we instantiated later on, yielding a (concrete) planning system.

$$\begin{aligned}
\mathsf{att}(\langle x_1,\ldots,x_k\rangle_k, \langle y_1,\ldots,y_k\rangle_k) &\to \mathsf{att}(x_i,y_i) \text{ with } i \le k\\
\mathsf{att}(\mathsf{senc}(x_1,x_2),\mathsf{senc}(y_1,y_2)),\ \mathsf{att}(x_2,y_2) &\to \mathsf{att}(x_1,y_1)\\
\mathsf{att}(\mathsf{aenc}(x_1,\mathsf{pub}(x_2)),\mathsf{aenc}(y_1,\mathsf{pub}(y_2))),\ \mathsf{att}(x_2,y_2) &\to \mathsf{att}(x_1,y_1)\\
\mathsf{att}(\mathsf{sign}(x_1,x_2),\mathsf{sign}(y_1,y_2)) &\to \mathsf{att}(x_1,y_1)
\end{aligned}$$

These rules correspond to the attacker's ability to project, decrypt, and retrieve messages from their signature. There is no Del since the attacker never forgets. Given a rule $r \in \mathsf{R}_{\mathsf{Ana}}$, we explain how to compute its concretization.

$\mathsf{Concrete}^+(r)$. The positive concretizations of r consist of instantiating r such that the resulting terms are messages. More formally, we have:

$$\mathsf{Concrete}^+(r) = \{r\sigma \mid \sigma \text{ substitution such that } r\sigma \text{ only involve messages.}\}$$

$\mathsf{Concrete}^-(r)$. We say that a sequence of facts $\mathsf{att}(u_1,v_1),\ldots,\mathsf{att}(u_k,v_k)$ left-unifies with a sequence $\mathsf{att}(u'_1,v'_1),\ldots,\mathsf{att}(u'_k,v'_k)$ if there exists σ such that $u'_1\sigma = u_1,\ldots,u'_k\sigma = u_k$ (and symmetrically for right-unification). Given an abstract attacker rule $r = \mathsf{Pre} \to \mathsf{Add}$, we define $\mathsf{Concrete}^-(r)$ as the set containing $f_1,\ldots,f_k \to \mathsf{bad}$ for any sequence of facts $f_1,\ldots,f_k \in \mathcal{F}act$ such that $f_1,\ldots,f_k$ left-unifies with Pre, whereas $f_1,\ldots,f_k$ does not right-unify with Pre.

Example 6. The negative concretizations of the abstract rule corresponding to asymmetric decryption are all the concrete rules of the form

$$\mathsf{att}(\mathsf{aenc}(u_1,\mathsf{pub}(u_2)),v),\ \mathsf{att}(u_2,v') \to \mathsf{bad}$$

where u_1,u_2,v,v', $\mathsf{aenc}(u_1,\mathsf{pub}(u_2))$ are messages, whereas $\mathsf{adec}(v,v')\!\downarrow$ is not.

3.3 Static inclusion

According Definition 4, to break static inclusion, an attacker may build new terms (using both analysis and synthesis rules) but also check for equalities and computation failures. To encode static inclusion using planning in an efficient way, we need to strictly control the terms that an attacker has to synthetise.

We say that R is *destructor-only* if $R \in \mathcal{T}(\Sigma_{\mathsf{d}}, \mathcal{C}_0 \cup \mathcal{W})$. It is *simple* if there exist destructor-only recipes $R_1,\ldots,R_k$, and a context C made of constructors such that $R = C[R_1,\ldots,R_k]$.

Definition 7. *Let ϕ,ψ be such that $dom(\phi) = dom(\psi)$. We write $\phi \sqsubseteq_s^{\mathsf{simple}} \psi$ if:*

1. *For each destructor-only recipe R such that $R\phi\!\downarrow$ is a (resp. atomic) message, $R\psi\!\downarrow$ is a (resp. atomic) message.*
2. *For each simple recipe R and destructor-only recipe R' such that $R\phi\!\downarrow, R'\phi\!\downarrow$ are messages and $R\phi\!\downarrow = R'\phi\!\downarrow$, we have that $R\psi\!\downarrow = R'\psi\!\downarrow$.*

3. *For each destructor-only recipes R, R', if $R\phi\downarrow = \mathsf{sign}(t, s)$, and $R'\phi\downarrow = \mathsf{vk}(s)$ for some term t and atom s, then $R\psi\downarrow = \mathsf{sign}(t', s')$, and $R'\psi\downarrow = \mathsf{vk}(s')$ for some term t' and atom s'.*
4. *For each destructor-only recipe R, such that $R\phi\downarrow = \mathsf{pub}(s)$ for atom s, $R\psi\downarrow = \mathsf{pub}(s')$ for some atom s'.*

We write $\phi \sqsubseteq_s^{\mathsf{simple}^+} \psi$ when the test described at item 2 is only performed when (i) either R is destructor-only; (ii) or $\mathsf{root}(R) \notin \{\mathsf{senc}\} \cup \{\langle\,\rangle_k \mid 2 \leq k \leq n\}$, and $\mathsf{root}(R') \neq \mathsf{adec}$.

This notion of static inclusion is equivalent to the original one.

Lemma 1. *Let ϕ and ψ be two frames having the same domain. We have that:*

$$\phi \sqsubseteq_s \psi \;\Leftrightarrow\; \phi \sqsubseteq_s^{\mathsf{simple}} \psi \;\Leftrightarrow\; \phi \sqsubseteq_s^{\mathsf{simple}^+} \psi.$$

From this new characterisation of static inclusion $\sqsubseteq_s^{\mathsf{simple}}$, we derive the planning rules that capture all the cases of failures.

$$\mathcal{R}_{\mathsf{fail}}^{\mathsf{atom}} = \{\mathsf{att}(u, v) \to \mathsf{bad} \mid u \text{ is an atom but } v \text{ is not}\}$$

$$\mathcal{R}_{\mathsf{fail}}^{\mathsf{pub}} = \{\mathsf{att}(\mathsf{pub}(u), v) \to \mathsf{bad} \mid v \text{ is not of the form } \mathsf{pub}(v')\}$$

$$\mathcal{R}_{\mathsf{fail}}^{\mathsf{check}} = \left\{ \begin{array}{l} \mathsf{att}(\mathsf{sign}(u_1, u_2), v_1) \\ \mathsf{att}(\mathsf{vk}(u_2), v_2) \end{array} \to \mathsf{bad} \mid \mathsf{check}(v_1, v_2)\downarrow \text{ is not a message} \right\}$$

$$\mathcal{R}_{\mathsf{fail}}^{\mathsf{test}} = \left\{ \begin{array}{l} \mathsf{att}(u_1, v_1), \ldots, \mathsf{att}(u_k, v_k) \\ \mathsf{att}(C[u_1, \ldots, u_k], v) \end{array} \to \mathsf{bad} \mid \begin{array}{l} C \text{ is a constructor context,} \\ C[u_1, \ldots, u_k] \in St(\phi) \cup \mathcal{C}_0 \\ v \neq C[v_1, \ldots, v_k]. \end{array} \right\}$$

Actually, not all subterms of $St(\phi)$ need to be considered. Therefore, we consider an optimised version that captures only the terms that may not be reconstructed from their subterms. Formally, $St_{\mathsf{opti}}(t)$ is defined as follows.

- $St_{\mathsf{opti}}(\langle t_1, t_2\rangle) = St_{\mathsf{opti}}(t_1) \cup St_{\mathsf{opti}}(t_2)$;
- $St_{\mathsf{opti}}(\mathsf{senc}(t_1, t_2)) = St_{\mathsf{opti}}(t_1)$;
- $St_{\mathsf{opti}}(\mathsf{aenc}(t_1, t_2)) = \{\mathsf{aenc}(t_1, t_2)\} \cup (St_{\mathsf{opti}}(t_1) \setminus \{t_1\})$
- $St_{\mathsf{opti}}(\mathsf{sign}(t_1, t_2)) = \{\mathsf{sign}(t_1, t_2) \cup St_{\mathsf{opti}}(t_1)$;
- $St_{\mathsf{opti}}(\mathsf{f}(t)) = \{\mathsf{f}(t)\}$ with $\mathsf{f} \in \{\mathsf{hash}, \mathsf{pub}, \mathsf{vk}\}$.

Thanks to the fact that $\sqsubseteq_s^{\mathsf{simple}^+}$ is equivalent to static inclusion, we may only consider simple recipes which evaluation yields a term in $St_{\mathsf{opti}}(\phi)$.

Lemma 2. *Let ϕ be a frame, $R = C[R_1, \ldots, R_k]$ be a simple recipe such that $\mathsf{root}(R) \notin \{\mathsf{senc}\} \cup \{\langle\,\rangle_k \mid 2 \leq k \leq n\}$, and R' be a destructor-only recipe such that $\mathsf{root}(R') \neq \mathsf{adec}$. Assume that $R\phi\downarrow$ and $R'\phi\downarrow$ are both messages such that $R\phi\downarrow = R'\phi\downarrow$. We have that either C is the empty context, or $R\phi\downarrow \in St_{\mathsf{opti}}(\phi) \cup \mathcal{C}_0$.*

Therefore, $\mathcal{R}_{\mathsf{fail}}^{\mathsf{test}}$ can be replaced by the following (smaller) set of rules:

$$\mathcal{R}_{\mathsf{fail}}^{\mathsf{test}_1} = \{\mathsf{att}(u_1, v_1), \mathsf{att}(u_1, v_2) \to \mathsf{bad} \mid v_1 \neq v_2\}$$

$$\mathcal{R}_{\mathsf{fail}}^{\mathsf{test}_2} = \{\mathsf{att}(u_1, v_1), \ldots, \mathsf{att}(u_k, v_k), \mathsf{att}(C[u_1, \ldots, u_k], v) \to \mathsf{bad} \mid C \text{ is a non-empty constructor context, } C[u_1, \ldots, u_k] \in St_{\mathsf{opti}}(\phi) \cup \mathcal{C}_0, \text{ and } v \neq C[v_1, \ldots, v_k].\}$$

Let ϕ and ψ be two frames with $dom(\phi) = dom(\psi)$ and built using constants from $\mathcal{C} \subseteq \mathcal{C}_0$. The set of facts associated to ϕ and ψ is defined as follows:

$$\mathsf{Fact}_{\mathcal{C}}(\phi, \psi) = \{\mathsf{att}(\mathsf{a}, \mathsf{a}) \mid \mathsf{a} \in \mathcal{C}\} \cup \{\mathsf{att}(\mathsf{w}\phi, \mathsf{w}\psi) \mid \mathsf{w} \in dom(\phi)\}$$

Two frames are in static inclusion if, and only if, the corresponding planning system has no solution. Actually, when the frames are not in static inclusion, we provide a bound on the length of the (minimal) plan witnessing this fact.

Proposition 1. *Let ϕ and ψ be two frames with $dom(\phi) = dom(\psi)$, and $\Theta = \langle \mathcal{F}act_0, \mathsf{Fact}_{\mathcal{C}_0}(\phi, \psi), \mathcal{R} \rangle$ where*

$$\mathcal{R} = \mathsf{Concrete}(\mathsf{R}_{\mathsf{Ana}}) \cup \mathcal{R}_{\mathsf{fail}}^{\mathsf{test}_1} \cup \mathcal{R}_{\mathsf{fail}}^{\mathsf{test}_2} \cup \mathcal{R}_{\mathsf{fail}}^{\mathsf{atom}} \cup \mathcal{R}_{\mathsf{fail}}^{\mathsf{check}} \cup \mathcal{R}_{\mathsf{fail}}^{\mathsf{pub}}.$$

Let $\Pi = \langle \Theta, \{\mathsf{bad}\} \rangle$. We have that $\phi \not\sqsubseteq_s \psi$ if, and only if, Π has a solution of length at most $(N+1) \times \mathsf{depth}(\phi) + 1$ where N is the number of names n occurring in ϕ at a key position, i.e. such that n (resp. $\mathsf{pub}(n)$) occurs in key position of an encryption in ϕ.

Intuitively, once all needed keys are derived, the minimal plan witnessing non-inclusion contains at most $\mathsf{depth}(\phi)$ rules where $\mathsf{depth}(\phi)$ is the maximal depth of a term occurring ϕ. Then we may need $\mathsf{depth}(\phi)$ rule to derive each deducible key, hence the bound.

4 From trace inclusion to planning

We are now ready for the active case. Given two configurations, we show how to build a planning problem such that the planning problem has a solution if, and only if, the two corresponding configurations are not in trace inclusion.

In several places of this section, we will consider three special constants, namely $c_0^\star$ and $c_1^\star$ of sort atom, and $c_+^\star$ of sort $\mathsf{bitstring}$. These three constants have a special type, denoted $\tau_\star$.

4.1 Abstract protocol rules

We first define the abstract rules describing the protocol behaviour. We denote $\mathcal{C}_\mathcal{P}$ (resp. $\mathcal{C}_\mathcal{Q}$) the constants from $\mathcal{C}_0$ occurring in $\mathcal{P}$ (resp. $\mathcal{Q}$), For simplicity we assume that variables of $\mathcal{P}$ and $\mathcal{Q}$ are disjoint. In addition to the facts of the form $\mathsf{att}(u, v)$ used to represent attacker's knowledge, we also consider:

- facts of the form $\mathsf{Phase}(i)$ with $i \in \mathbb{N}$ to represent phases; and
- facts of the form $\mathsf{St}(P, Q) = \mathsf{state}^c_{P,Q}(id_P, id_Q)$ where P, Q are two basic processes on channel c, and id_P (resp. id_Q) is the identity substitution of domain $fv(P)$ (resp. $fv(Q)$).

Therefore, in this section, we consider the infinite set of facts $\mathcal{F}act_0$ that consists of all the ground facts of this form, plus the special symbol bad.

To deal with phases, we mimic the PHASE rule by considering basic processes in normal form w.r.t. the rule $i{:}j{:}P \to j{:}P$. Then, the transformation $\mathsf{Rule}(P; Q)$ from basic processes (in normal form) to abstract planning rules is defined by $\mathsf{Rule}(P; Q) = \emptyset$ when $P = i{:}0$, and otherwise:

1. Case output: i.e. if $P = i{:}\mathsf{out}(c,u).P'$.
   - $\{\mathsf{St}(P,Q), \mathsf{Phase}(i) \to \mathsf{att}(u,v), \mathsf{St}(P',Q');\ \mathsf{St}(P,Q)\} \cup \mathsf{Rule}(i{:}P'; i{:}Q')$ when if $Q = i{:}\mathsf{out}(c,v).Q'$
   - $\{\mathsf{St}(P,Q), \mathsf{Phase}(i) \to \mathsf{att}(u,c_0^\star),\ \mathsf{bad}\}$ otherwise.
2. Case input: i.e. $P = i{:}\mathsf{in}(c,u).P'$.
   - $\{\mathsf{St}(P,Q), \mathsf{att}(u,v), \mathsf{Phase}(i) \to \mathsf{St}(P',Q');\ \mathsf{St}(P,Q)\} \cup \mathsf{Rule}(i{:}P'; i{:}Q')$ when $Q = i{:}\mathsf{in}(c,v).Q'$
   - $\{\mathsf{St}(P,Q), \mathsf{att}(u,x), \mathsf{Phase}(i) \to \mathsf{bad}\}$ otherwise (with x fresh).

Intuitively, abstract rules simply try to mimic each step of P by a similar step in Q. Clearly, if Q cannot follow P, the two processes are not in trace equivalence, which is modelled here by the bad state. Note that, in case $P = i{:}\mathsf{out}(c,u).P'$ whereas Q is not ready to perform an output, bad will be triggered only if the sent term is indeed a message. This transformation is then extended to protocols in a natural way considering in addition planning rule to model phase changes. We consider $\mathcal{P} = \{P_1, \ldots, P_n\}$ and $\mathcal{Q} = \{Q_1, \ldots, Q_n\}$, and we assume w.l.o.g. that P_i and Q_i are basic processes on channel c_i. We define:

- $\mathsf{Rule}(\mathcal{P}, \mathcal{Q}) = \mathsf{Rule}(P_1, Q_1) \cup \ldots \cup \mathsf{Rule}(P_n, Q_n)$.
- $\mathcal{R}^{\mathsf{phase}} = \{\mathsf{Phase}(i) \to \mathsf{Phase}(i+1)\,;\, \mathsf{Phase}(i) \mid i \in \mathbb{N}\}$.

4.2 Concrete protocol rules

Deriving concrete rules from the abstract ones can be obtained by instantiating them with arbitrary terms. However, this would not allow us to derive a decision procedure. Moreover, we would like our algorithm to have good performance. To achieve this, we first show that only three constants need to be considered (and no nonces), in addition to those explicitly mentioned in the protocol.

Given a protocol $\mathcal{P}$ that is type-compliant w.r.t. to a typing system $(\mathcal{T}_\mathcal{P}, \delta_\mathcal{P})$ (and such that $\tau_\star$ does not occur in $\delta_\mathcal{P}(\mathcal{P})$), an execution $\mathcal{P} \xrightarrow{\mathsf{tr}} (\mathcal{P}'; \phi'; \sigma'; i')$ is *quasi-well-typed* if $\delta_\mathcal{P}(x\sigma') \preceq \delta_\mathcal{P}(x)$ for every variable $x \in dom(\sigma')$ where $\preceq$ is the smallest relation on types defined as follows:

- $\tau_\star \preceq \tau$ and $\tau \preceq \tau$ for any type τ (initial or not);
- $\mathsf{f}(\tau_1, \ldots, \tau_k) \preceq \mathsf{f}(\tau'_1, \ldots, \tau'_k)$ when $\tau_1 \preceq \tau'_1, \ldots, \tau_k \preceq \tau'_k$, and $\mathsf{f} \in \Sigma_\mathsf{c}$.

The attacker needs at most the constants $c_0^\star, c_1^\star, c_+^\star$ to mount an attack.

Theorem 1. *Let $\mathcal{K}_\mathcal{P}$ be an initial $\mathcal{C}_0$-configuration type-compliant w.r.t. $(\mathcal{T}_\mathcal{P}, \delta_\mathcal{P})$ and $\mathcal{K}_\mathcal{Q}$ be another initial $\mathcal{C}_0$-configuration. Let $\mathcal{C}^\star = (\mathcal{C}_\mathcal{P} \cup \mathcal{C}_\mathcal{Q}) \uplus \{c_0^\star, c_1^\star, c_+^\star\}$. We have that $\mathcal{K}_\mathcal{P} \not\sqsubseteq_t \mathcal{K}_\mathcal{Q}$ w.r.t. $\mathcal{C}_0$ if, and only if, there exists a witness $(\mathsf{tr}, \phi) \in \mathsf{trace}(\mathcal{K}_\mathcal{P})$ of this non-inclusion which only involves constants from $\mathcal{C}^\star$, simple recipes, and with a quasi-well-typed underlying execution.*

The existence of a quasi well-typed witness comes from [17] with some extra work to guarantee that we can consider simple recipes. The reduction to three constants extends the previous reduction [20] to asymmetric primitives.

Flattening. In terms of efficiency, one key step of our algorithm is to avoid composition rules from the attacker. For static inclusion, we only consider specific contexts, hence very specific synthesis rules, guided by the form of the underlying frames. For the active case, we transform protocol rules in order to pre-compute all necessary composition steps. This flattening step was already used in e.g. [4, 20], and is quite intuitive. We therefore only describe it informally on an example.

Consider our Denning Sacco protocol presented in Example 1. Agent B expects a message of the form $u = \{\mathsf{sign}(\langle \mathsf{a}, \mathsf{b}, x\rangle_3, sk_a)\}_{\mathsf{pub}(sk_b)}$. Either the attacker obtains a message m of the expected form, or the attacker obtains several components of it and forges the whole message. For example, it is sufficient for him to obtain m_1 of the form $u_1 = \mathsf{sign}(\langle \mathsf{a}, \mathsf{b}, x\rangle_3, sk_a)$ and m_2 of the form $u_2 = \mathsf{pk}(sk_b)$. Therefore, in addition to the (informal) protocol rule $u \to \ldots$, we also consider the rule $u_1, u_2 \to \ldots$. Similarly, we also need to consider the rules $\mathsf{a}, \mathsf{b}, x, sk_a, \mathsf{pk}(sk_b) \to \ldots$ and $\mathsf{a}, \mathsf{b}, x, sk_a, sk_b \to \ldots$.

More generally, given an abstract protocol rule r, we can define $\mathsf{Flat}(r)$ the set of rules obtained by performing flattening on each fact. To decompose a term, we follow its structure, and the structure of a variable is given by its type. Moreover, when the other side of the process is not able to follow the decomposition, this leads us to a failure rule. A formal definition is provided in Appendix D.

Similarly to the static case, we define $\mathsf{Concrete}^+(r)$ as the set of ground, quasi well-typed instantiations of a rule r where the instantiation of a state is obtained by simply composing the substitutions, i.e.

$$st^c_{P,Q}(\sigma_P, \sigma_Q)\sigma = st^c_{P,Q}(\sigma \circ \sigma_P, \sigma \circ \sigma_Q).$$

We also define $\mathsf{Concrete}^-(r)$ the concrete rule yielding to the bad state, when the right-hand side of the rule does not unify or is not a message. Given a pair of protocols, we initialise our planning system as follows.

$$\begin{aligned}\mathsf{Fact}_{\mathcal{C}}(\mathcal{P}, \mathcal{Q}) = {} & \{\mathsf{Phase}(0)\} \cup \{\mathsf{att}(c, c) \mid c \in \mathcal{C}\} \cup \\ & \{\mathsf{state}^c_{P,Q}(\emptyset, \emptyset) \mid P \in \mathcal{P}, Q \in \mathcal{Q} \text{ basic processes built on } c\}.\end{aligned}$$

Our main technical result states that our encoding is sound and complete: two protocols are in trace inclusion if, and only if, the corresponding planning system has a solution. Moreover, when a witness of non-inclusion exists, we are able to bound the length of the resulting plan. Below, $\mathsf{nb}_{\mathsf{in}}(\mathcal{P})$ (resp. $\mathsf{nb}_{\mathsf{out}}(\mathcal{P})$) denotes the number of inputs (resp. outputs) occurring in $\mathcal{P}$ whereas $\mathsf{max}_{\mathsf{phase}}(\mathcal{P})$ is the maximal integer occurring in a phase instruction in $\mathcal{P}$.

Theorem 2. *Let $\mathcal{P}$ a protocol type-compliant w.r.t. $(\mathcal{T}_{\mathcal{P}}, \delta_{\mathcal{P}})$, and $\mathcal{Q}$ be another protocol. We consider the following set $\mathcal{R}$ of concrete rules:*

$$\mathsf{Concrete}(\mathsf{R}_{\mathsf{Ana}} \cup \mathsf{Flat}(\mathsf{Rule}(\mathcal{P}, \mathcal{Q}))) \cup \mathcal{R}^{\mathsf{phase}} \cup \mathcal{R}^{\mathsf{test}_1}_{\mathsf{fail}} \cup \mathcal{R}^{\mathsf{test}_2}_{\mathsf{fail}} \cup \mathcal{R}^{\mathsf{atom}}_{\mathsf{fail}} \cup \mathcal{R}^{\mathsf{check}}_{\mathsf{fail}} \cup \mathcal{R}^{\mathsf{pub}}_{\mathsf{fail}}$$

Let $\Theta = \langle \mathcal{F}act_0, \mathsf{Fact}_{\mathcal{C}^\star}(\mathcal{P}, \mathcal{Q}), \mathcal{R}\rangle$ and $\Pi = \langle \Theta, \{\mathsf{bad}\}\rangle$. We have that $\mathcal{P} \not\sqsubseteq_t \mathcal{Q}$ if, and only if, Π has a solution of length

$$1 + \mathsf{nb}_{\mathsf{in}}(\mathcal{P}) + \mathsf{nb}_{\mathsf{out}}(\mathcal{P}) + \mathsf{max}_{\mathsf{phase}}(\mathcal{P}) + \mathsf{depth}(\delta_{\mathcal{P}}(\mathcal{P})) \times [1 + \mathsf{nb}_{\mathsf{in}}(\mathcal{P}) + N]$$

where N is the number of names occurring in $\mathcal{P}$ having a key type, i.e. such that $\delta_{\mathcal{P}}(n)$ (resp. $\mathsf{pub}(\delta_{\mathcal{P}}(n))$) occurs in key position of an encryption in $\delta_{\mathcal{P}}(\mathcal{P})$.

Proof. (Sketch) It is rather easy to establish that a solution to the planning problem defines a witness of non trace inclusion, Conversely, thanks to Theorem 1, if $\mathcal{P} \not\sqsubseteq_t \mathcal{Q}$, then there exists a quasi well-typed witness of non trace inclusion, that uses at most three constants (besides the constants of $\mathcal{P}$ and $\mathcal{Q}$). This witness guides the definition of a plan of Π. Establishing a not too coarse bound on its length requires some care. It relies on the flattening of the protocol and the fact that the plan can mimic the computation of several messages in parallel.

5 Algorithm

Similarly to the algorithm presented in [20], we decide trace inclusion by applying graph planning and SAT-solving techniques to the planning problem that encodes trace inclusion (thanks to Theorem 2). Given a protocol $\mathcal{P}$, type-compliant w.r.t. $(\mathcal{T}_\mathcal{P}, \delta_\mathcal{P})$, and a protocol $\mathcal{Q}$, our algorithm proceeds as follows.

1. It first computes the corresponding abstract rules, namely $\mathsf{Flat}(\mathsf{Rule}(\mathcal{P}; \mathcal{Q})) \cup \mathsf{R}_{\mathsf{Ana}}$ and the initial state $\mathsf{Fact}(\mathcal{P}, \mathcal{Q})$.
2. It then applies a planning graph algorithm, a standard technique to solve planning problems (see e.g. [9]). The only difference is that, for efficiency reasons, we do not contruct the planning problem Π *a priori* but instead, we compute it "on the fly", while building the associated planning graph. This planning graph over-approximates the possible solutions by executing several actions in parallel, even if they may be incompatible. Some incompatibilities are recorded and propagated through so-called mutex. The planning graph is deemed to capture all possible plans. More precisely, the planning graph built until depth k captures all possible plans of length at most k.
3. In case no fact bad has been reached while building the planning graph, we can immediately conclude that $\mathcal{P} \sqsubseteq_t \mathcal{Q}$. Otherwise, since the planning graph over-approximates the possible executions, we need to check that bad is truly reachable. This is done by encoding each path leading to bad as a SAT formula. We then call the SAT solver mini-SAT to decide its satisfiability. In case bad is indeed reachable, mini-SAT provides a solution that is translated back to a witness of non-inclusion. To improve termination, we check accessibility of a state containing bad as soon as it appears in the graph, even if the construction of the graph is not completed yet.

Termination. The algorithm defined above may not terminate. The planning graph contains facts of the form $\mathsf{att}(u, v)$ where u must be (quasi) well-typed. There is therefore only a finite number of such u. However, the planning graph construction may introduce several facts of the form $\mathsf{att}(u, v_1), \ldots, \mathsf{att}(u, v_k)$, where the v_i get arbitrarily large. We exhibit some (contrived) examples in Appendix E where the algorithm does not terminate. [20] suggests that termination could be enforce by checking at each step (thanks to the SAT-solver) that each node of the planning graph is indeed reachable. This would be however not practical. Instead, we can enforce termination thanks to the bound provided in

	Spec	Akiss	Deepsec	CSF'17	Sat-Eq
Denning-Sacco	7	10	35	98	> 210 (4h)
Needham-Schroeder sym	6	6	21	21	94^* (20h30)
Wide Mouth Frog	7	12	28	84	> 210 (6min)
Yahalom-Paulson	6	6	12	7	> 28 (7h)
Passive Authentication	6	8	46	–	> 400 (98s)
Active Authentication	6	8	50	–	> 400 (78s)
Needham-Schroeder-Lowe	4	6	16	–	> 64 (11min)
Denning-Sacco signature	8	8	18	–	> 64 (100s)

Fig. 2. Comparison of SAT-Equiv with the other tools. We indicate the number of sessions for which the tool fails (time out, memory out, or other issues). When we did not reach the limit of the tool, we write $\geq k$ to indicate that the tool can analyse more than k sessions, and we indicate the analysis time for k. * see Section 6.2

Theorem 2 that also bounds the maximal depth of the planning graph that needs to be considered. Indeed, it is sufficient to simply stop the construction of the planning graph as soon as the bound is reached. The interest of this approach is that we guarantee termination at no cost (computing the bound is immediate). In practice, the planning graph is typically much smaller than this bound.

SAT-Equiv. We have implemented our new algorithm in the tool SAT-Equiv, extending it to protocols with phases and all the standard cryptographic primitives and guaranteeing termination. Moreover, we significantly improve its efficiency by rewriting parts of the codes and modifying the data structure.

6 Experiments

In this section, we analyse several protocols of the literature and compare the results obtained using different tools. We ran our experiments a single Intel 3.1 GHz Xeon. We limit the memory to 128 Go (MO stands for memory out) and the execution time to 24h (TO stands for time out).

For all the considered protocols, we analyse strong secrecy of the exchanged key or nonce, as for Example 5, except for the passport protocol (PA), for which we prove anonymity as in [3]. We progressively increase the number of sessions in order to consider *a semi complete scenario*, where Alice's role is instantiated by honest a talking to honest b or dishonest c and Bob's role is instantiated by b talking to a or c. This typically corresponds to 7 sessions in the case of a symmetric key protocol (with 3 roles).

6.1 Comparison with the other tools

Our experiments show a significant speed-up w.r.t. the original version of SAT-Equiv [20]. Our new is 100 faster in average, allowing to analyse about twice more sessions, as exemplified in Figure 2 (more experiments in [1]). We compare SAT-Equiv with other tools of the literature that decide equivalence for a bounded

number of sessions, namely Spec [24], Akiss [10] and Deepsec [13]. We did not include APTE in our study [11] as it is now subsumed by Deepsec. For each protocol, we progressively increased the number of sessions until we reached a time out. The overall results of our experiments are summarized in Figure 2. They show a significant speed-up even w.r.t. the very recent Deepsec tool. Note however that Deepsec covers more protocols (with else branches, or not type compliant), except if they include phases. Deepsec can also be parallelized thus the analysis time can be divided by the number of available cores. The detailed results for the Denning-Sacco protocol are below.

Denning-Sacco	Spec	Akiss	Deepsec	CSF'17	SAT-Equiv	
3	12 s	0.08 s	<0.01 s	0.3 s	0.07 s	42
6	5 h	9 s	<0.01 s	1 s	0.1 s	64
7	MO	75 s	<0.01 s	2 s	0.2 s	74
10		MO	0.01 s	4 s	0.3 s	114
21			18 s	60 s	1.3 s	216
35			TO	9 min	6 s	344
84				13 h	164 s	792
98				TO	6 min	920
210					4h20	1942

The 2nd column for SAT-Equiv indicates the theoretical bound on the length of the planning graph, as given by Theorem 2. This illustrates that this bound remains reasonable although our tool actually terminates before reaching it.

6.2 Towards an unbounded number of sessions

Although equivalence is undecidable in general for an unbounded number of sessions, [16] exhibits a decidability result, for type-compliant protocols that have an *acyclic* dependency graph. Intuitively, the dependency graph captures how a message expected as input may be built (and therefore may depend) from messages sent as output of the protocol. Decidability is proven by showing that a (minimal) attack trace may be mapped to this dependency graph. Looking at the dependency graphs of the Denning-Sacco and the Needham-Schroeder symmetric key protocols, we deduce that it is sufficient to analyse respectively 42 and 94 sessions. Thanks to the efficiency of SAT-Equiv, we can easily analyse 42 sessions of Denning-Sacco (in 10s). We can therefore deduce from [16] that the protocol remains secure even if the considered sessions are arbitrarily replicated. The case of the Needham-Schroeder protocol requires a bit more work as 94 sessions is slightly out of reach of SAT-Equiv. However, we noticed that, according to [16], we do not need to analyse 94 full sessions. Instead, some of some of them may be truncated (a minimal attack will use only the first step for example). Since SAT-Equiv can prove equivalence of these refined 94 sessions (in 20h30min), we can again deduce from [16] that the protocol remains secure even if the considered sessions are arbitrarily replicated.

As future work, we plan to optimize the bound on sessions induced by [16] and automatically generate the desired scenario, in order to extend SAT-Equiv to proofs of equivalence for an unbounded number of sessions.

References

1. `https://projects.lsv.ens-cachan.fr/satequiv`.
2. Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *Proceedings of the 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '01, pages 104–115, New York, NY, USA, 2001. ACM.
3. Myrto Arapinis, Tom Chothia, Eike Ritter, and Mark Ryan. Analysing unlinkability and anonymity using the applied pi calculus. In *23rd Computer Security Foundations Symposium (CSF'10)*, pages 107–121. IEEE Computer Society Press, 2010.
4. Alessandro Armando and Luca Compagna. Sat-based model-checking for security protocols analysis. *International Journal of Information Security*, 7:3–32, 2008.
5. Karthikeyan Bhargavan, Bruno Blanchet, and Nadim Kobeissi. Verified models and reference implementations for the TLS 1.3 standard candidate. In *IEEE Symposium on Security and Privacy 2017*, pages 483–502, 2017.
6. B. Blanchet. *Vérification automatique de protocoles cryptographiques : modèle formel et modèle calculatoire. Automatic verification of security protocols: formal model and computational model.* Mémoire d'habilitation à diriger des recherches, Université Paris-Dauphine, November 2008.
7. Bruno Blanchet. Symbolic and computational mechanized verification of the arinc823 avionic protocols. In *30th IEEE Computer Security Foundations Symposium (CSF'17)*, pages 68–82. IEEE, 2017.
8. Bruno Blanchet, Martín Abadi, and Cédric Fournet. Automated Verification of Selected Equivalences for Security Protocols. *Journal of Logic and Algebraic Programming*, 75(1):3–51, February–March 2008.
9. Avrim Blum and Merrick Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90:281–300, 1997.
10. Rohit Chadha, Ştefan Ciobâcă, and Steve Kremer. Automated verification of equivalence properties of cryptographic protocols. In *21th European Symposium on Programming (ESOP'12)*, LNCS, 2012.
11. Vincent Cheval. Apte: an algorithm for proving trace equivalence. In Erika Ábrahám and JKlaus Havelund, editors, *Proceedings of the 20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'14)*, volume 8413 of *Lecture Notes in Computer Science*, pages 587–592, Grenoble, France, April 2014. Springer Berlin Heidelberg.
12. Vincent Cheval, Véronique Cortier, and Stéphanie Delaune. Deciding equivalence-based properties using constraint solving. *Theoretical Computer Science*, 492:1–39, June 2013.
13. Vincent Cheval, Steve Kremer, and Itsaka Rakotonirina. Deepsec: Deciding equivalence properties in security protocols - theory and practice. In *39th IEEE Symposium on Security and Privacy (S&P'18)*, pages 525–542. IEEE Computer Society Press, 2018.
14. Rémy Chrétien, Véronique Cortier, and Stéphanie Delaune. Typing messages for free in security protocols: the case of equivalence properties. In Paolo Baldan and Daniele Gorla, editors, *Proceedings of the 25th International Conference on Concurrency Theory (CONCUR'14)*, volume 8704 of *Lecture Notes in Computer Science*, pages 372–386, Rome, Italy, September 2014. Springer.
15. Rémy Chrétien, Véronique Cortier, and Stéphanie Delaune. Checking trace equivalence: How to get rid of nonces? In *Proceedings of the 20th European Symposium*

on Research in Computer Security (ESORICS'15), Lecture Notes in Computer Science, Vienna, Austria, 2015. Springer.

16. Rémy Chrétien, Véronique Cortier, and Stéphanie Delaune. Decidability of trace equivalence for protocols with nonces. In *Proceedings of the 28th IEEE Computer Security Foundations Symposium (CSF'15)*, pages 170–184, Verona, Italy, July 2015. IEEE Computer Society Press.
17. Rémy Chrétien, Véronique Cortier, Antoine Dallon, and Stéphanie Delaune. Typing messages for free in security protocols. Technical report, 2018.
18. Luca Compagna. *SAT-based Model-Checking of Security Protocols*. PhD thesis, Università degli Studi di Genova and the University of Edinburgh (joint programme), September 2005.
19. Véronique Cortier, Antoine Dallon, and Stéphanie Delaune. Bounding the number of agents, for equivalence too. In Frank Piessens and Luca Viganó, editors, *Proceedings of the 5th International Conference on Principles of Security and Trust (POST'16)*, Lecture Notes in Computer Science, Eindhoven, The Netherlands, April 2016. Springer.
20. Véronique Cortier, Antoine Dallon, and Stéphanie Delaune. SAT-Equiv: an efficient tool for equivalence properties. In *Proceedings of the 30th IEEE Computer Security Foundations Symposium (CSF'17)*, Santa Barbara, CA, USA, August 2017. IEEE Computer Society Press.
21. Véronique Cortier, Niklas Grimm, Joseph Lallemand, and Matteo Maffei. A type system for privacy properties. In *24th ACM Conference on Computer and Communications Security (CCS'17)*, pages 409–423. ACM, 2017.
22. Véronique Cortier, Niklas Grimm, Joseph Lallemand, and Matteo Maffei. Equivalence properties by typing in cryptographic branching protocols. In *Proceedings of the 7th International Conference on Principles of Security and Trust (POST'18)*, LNCS, 2018.
23. Cas Cremers, Marko Horvat, Jonathan Hoyland, Sam Scott, and Thyla van der Merwe. A comprehensive symbolic analysis of TLS 1.3. In *24th ACM Conference on Computer and Communications Security (CCS'17)*, pages 1773–1788, 2017.
24. Jeremy Dawson and Alwen Tiu. Automating open bisimulation checking for the spi-calculus. In *Proceedings of IEEE Computer Security Foundations Symposium (CSF 2010)*, 2010.
25. Stéphanie Delaune, Steve Kremer, and Mark Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, (4):435–487, July 2008.
26. Santiago Escobar, Catherine Meadows, and José Meseguer. A rewriting-based inference system for the NRL protocol analyzer and its meta-logical properties. *Theoretical Computer Science*, 367(1-2):162–202, 2006.
27. Henri Kautz and Bart Selman. Planning as satisfiability. In *ECAI*, pages 359–363, Vienna, Austria, 1992.
28. Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification, 25th International Conference, CAV 2013, Princeton, USA, Proc.*, volume 8044 of *Lecture Notes in Computer Science*, pages 696–701. Springer, 2013.

A From static inclusion to planning

The main goal of this section is to prove Proposition 1. We introduce the notion of *forced normal form*, denoted $u\downarrow$. This is the normal form obtained when applying rewrite rules as soon as the destructor and the constructor match. More formally, we have that:

$$\mathcal{R}_f = \begin{cases} \mathsf{sdec}(\mathsf{senc}(x,y),z) \twoheadrightarrow x & \mathsf{adec}(\mathsf{aenc}(x,y),z) \twoheadrightarrow x \\ \mathsf{getmsg}(\mathsf{sign}(x,y)) \twoheadrightarrow x & \mathsf{check}(x,y) \twoheadrightarrow \mathsf{ok} \\ \mathsf{proj}_j^i(\langle x_1,\ldots,x_i\rangle_i) \twoheadrightarrow x_j & \text{with } 2 \leq i \leq n \text{ and } 1 \leq j \leq i \end{cases}$$

A term u can be rewritten in v using $\mathcal{R}_f$ if there is a position p in u, and a rewriting rule $\mathsf{g}(t_1,\ldots,t_n) \twoheadrightarrow t$ such that $u|_p = \mathsf{g}(t_1,\ldots,t_n)\theta$ for some substitution θ, and $v = u[t\theta]_p$. As usual, we denote $\twoheadrightarrow^*$, the reflexive-transitive closure of $\twoheadrightarrow$. We may note that such a rewriting system is confluent as it terminates and has no critical pair. From [17], we get the following Lemma:

Lemma 3. *Let ϕ be a frame, R be a recipe such that $R\phi\downarrow$ is a message, and R' be such that $R \twoheadrightarrow R'$. We have that R' is a recipe, and $R'\phi\downarrow = R\phi\downarrow$.*

Lemma 4. *Let R be a recipe in normal form w.r.t. $\twoheadrightarrow$ such that $R\phi\downarrow$ is a message for some frame ϕ. We have that R is a simple recipe.*

Proof. We prove this result by structural induction on R.

Base case: $R \in \mathcal{W} \cup \mathcal{C}_0$. In both cases, the result holds since R is a simple recipe.

Induction case: We have that $R = \mathsf{f}(R_1,\ldots,R_k)$ for some $\mathsf{f} \in \Sigma$, and we know that $R_1,\ldots,R_k$ are in normal form w.r.t. $\twoheadrightarrow$.

- *Case* $\mathsf{f} \in \Sigma_\mathsf{c}$. We have that $R\phi\downarrow = \mathsf{f}(R_1\phi\downarrow,\ldots,R_k\phi\downarrow)$ is a message for some frame ϕ, and thus $R_i\phi\downarrow$ is a message for $i \in \{1,\ldots,k\}$. Applying our induction hypothesis on R_i $(1 \leq i \leq k)$, we easily conclude.
- *Case* $\mathsf{f} = \mathsf{des} \in \Sigma_\mathsf{d} \uplus \{\mathsf{check}\}$. Let $\ell_\mathsf{des} \to r_\mathsf{des}$ be the rewriting rule associated to des, and $\ell'_\mathsf{des} \twoheadrightarrow r_\mathsf{des}$ be its associated forced rewriting rule. The case where $\mathsf{des} = \mathsf{check}$ is impossible as R is in forced normal form. Therefore, we have that $\mathsf{f} \in \{\mathsf{proj}_j^i, \mathsf{sdec}, \mathsf{adec}, \mathsf{getmsg}\}$. From now on, we assume that $\mathsf{f} = \mathsf{adec}$. The other cases can be done in a rather similar way.
  Since $\mathsf{f} = \mathsf{adec}$, we have that $R = \mathsf{f}(R_1,R_2)$. As $R\phi\downarrow$ is a message, $R_1\phi\downarrow$ and $R_2\phi\downarrow$ are messages. Applying our induction hypothesis, we deduce that both R_1 and R_2 are simple. As $R_2\phi\downarrow$ is an atomic message, we know that R_2 is destructor-only. First, we assume that $R_1 = \mathsf{g}(R'_1,\ldots,R'_n)$ for some $\mathsf{g} \in \Sigma_\mathsf{c}$. As $R\phi\downarrow$ is a message, $\mathsf{g} = \mathsf{aenc}$. However, this contradicts the fact that R is in forced normal form. Thus, R_1 is destructor-only, and therefore R is simple. □

Lemma 1. *Let ϕ and ψ be two frames having the same domain. We have that:*

$$\phi \sqsubseteq_s \psi \;\Leftrightarrow\; \phi \sqsubseteq_s^{\mathsf{simple}} \psi \;\Leftrightarrow\; \phi \sqsubseteq_s^{\mathsf{simple}^+} \psi.$$

Proof. It is easy to see that $\phi \sqsubseteq_s \psi \;\Rightarrow\; \phi \sqsubseteq_s^{\mathsf{simple}} \psi \;\Rightarrow\; \phi \sqsubseteq_s^{\mathsf{simple}^+} \psi$. Thus, we only consider the two other implications.

First implication: $\phi \sqsubseteq_s^{\mathsf{simple}} \psi \;\Rightarrow \phi \sqsubseteq_s \psi$. We consider an alternative definition of static inclusion, denoted $\sqsubseteq'_s$. This notion is similar to the one given in Definition 7 but considering arbitrary recipes instead of simple/destructor-only recipes. Clearly, we have that $\phi \sqsubseteq'_s \psi \;\Rightarrow \phi \sqsubseteq_s \psi$, and thus to conclude, it remains to establish $\phi \sqsubseteq_s^{\mathsf{simple}} \psi \;\Rightarrow \phi \sqsubseteq'_s \psi$.

More precisely, given an arbitrary test T that holds in ϕ, we show that T also holds in ψ assuming that any test smaller than T have already been transfered from ϕ to ψ. We consider the following measure where $|R|$ is simply the size of R, i.e. the number of function symbols occurring in it.

1. T is a recipe (message/atomic message/public key): $\mu(T) = |R|$
2. T is made of two recipes (equality test/signature test): $\mu(T) = |R| + |R'|$.

R is a recipe such that $R\phi\!\downarrow$ is a message (resp. atomic message).

- Case where R is *not* in normal form w.r.t. $\twoheadrightarrow$. Consider R' such that $R \twoheadrightarrow R'$. We have that $R'\phi\!\downarrow$ is a message (Lemma 3), By induction hypothesis $R'\psi\!\downarrow$ is a message too. It remains to show that $R\psi\!\downarrow$ is a message. Actually, we have that $R = C[\mathsf{adec}(\mathsf{aenc}(R_1, R_2), R_3)]$ and $R' = C[R_1]$ (other cases are similar). Since $R\phi\!\downarrow$ is a message, we know that $R_2\phi\!\downarrow = \mathsf{pub}(R_3)\phi\!\downarrow$. By induction hypothesis $R_2\psi\!\downarrow = \mathsf{pub}(R_3)\psi\!\downarrow$, and this allows us to conclude.
- Case where R is in normal form w.r.t. $\twoheadrightarrow$. In this case, we know that R is simple (Lemma 4), i.e. $R = C[R_1, \ldots, R_k]$, where C is a constructor context and R_i are destructor-only recipes. If C is empty, then R is destructor-only. We conclude by relying on our hypothesis. Otherwise $R = \mathsf{f}(R'_1, \ldots, R'_n)$. By induction hypothesis, we know that $R'_i\psi\!\downarrow$ is a message ($1 \leq i \leq n$). We have to prove that $C[R_1, \ldots, R_k]\psi\!\downarrow\mathsf{f}(R'_1, \ldots, R'_n)\psi\!\downarrow$ is a message. We have atomic messages at key positions (thanks to our induction hypothesis). In case $\mathsf{f} = \mathsf{aenc}$ (and thus $n = 2$), we have to ensure that $R'_2\psi\!\downarrow$ is of the form $\mathsf{pub}(s)$. This is given by item 4 of Definition 7.

R and R' are recipes, $R\phi\!\downarrow$, $R'\phi\!\downarrow$ are messages, and $R\phi\!\downarrow = R'\phi\!\downarrow$.

- Case R (resp. R') is *not* in normal form w.r.t. $\twoheadrightarrow$. Let $R'' = R\!\downarrow$. Since $R\phi\!\downarrow$ and $R\psi\!\downarrow$ are messages, we deduce that $R''\phi\!\downarrow = R\phi\!\downarrow$ and $R''\psi\!\downarrow = R\psi\!\downarrow$. We have that $R''\phi\!\downarrow = R\phi\!\downarrow = R'\phi\!\downarrow$. Relying on our induction hypothesis applied on the test $R'' = R'$, we deduce that $R''\psi\!\downarrow = R'\psi\!\downarrow$, and thus $R\psi\!\downarrow = R'\psi\!\downarrow$.
- Case R and R' are simple, i.e. $R = C[R_1, \ldots, R_k]$ and $R' = C'[R'_1, \ldots, R'_\ell]$, where C, C' are constructor contexts and R_i ($1 \leq i \leq k$) as well as R'_j ($1 \leq j \leq \ell'$) are destructor-only recipes. If neither C nor C' is empty (that is, neither R nor R' is destructor-only) then $\mathsf{root}(R) = \mathsf{root}(R')$, and thus we conclude relying on our induction hypothesis. Otherwise, we conclude relying on our hypothesis.

R and R' are recipes, $R\phi\downarrow = \mathsf{sign}(t,s)$, and $R'\phi\downarrow = \mathsf{vk}(s)$ for some term t and some atom s.

- Case R (resp. R') is *not* in forced normal form. $R\downarrow$ (resp. $R'\downarrow$) is a smaller recipe then R (resp. R'). By Lemma 3, $R\downarrow\phi\downarrow = R\phi\downarrow$ (resp. $R'\downarrow\phi\downarrow = R'\phi\downarrow$). So $R\downarrow, R'$ (resp. $R, R'\downarrow$) gives us a smaller test than R, R'. By induction hypothesis we get that say $R\downarrow\psi\downarrow = \mathsf{sign}(t',s')$ and $R'\psi\downarrow = \mathsf{vk}(s')$ for some term t and atom s. We already considered the case where $R\phi\downarrow$ is a message, so we can assume $R\psi\downarrow$ is a message. Then we deduce that $R\psi\downarrow = R\downarrow\psi\downarrow$ by Lemma 3, and it concludes this case.
- Case R and R' are both simple recipes. In case they are both destructor-only recipes we conclude relying on our hypothesis. Otherwise, assume first $R = \mathsf{sign}(R_1, R_2)$. In such a case, we have that $\mathsf{vk}(R_2) = R'$ and this test is smaller than $R = R'$, it holds in ϕ, and thus it can be transfered from ϕ to ψ by induction hypothesis. This allows us to conclude that $R = R'$ holds in ψ. Now, assume that $R' = \mathsf{vk}(R'_1)$. Since $R'_1\phi\downarrow$ is an atomic message, R'_1 is destructor only. We have that R' is destructor only, and thus $\mathsf{sign}(\mathsf{getmsg}(R), R')$ is simple, R is destructor-only and the $\mathsf{sign}(\mathsf{getmsg}(R), R'_1) = R$ holds in ϕ. By hypothesis, it also holds in ψ. This allows us to conclude that $R\psi\downarrow = \mathsf{sign}(t',s')$ with $R'_1\psi\downarrow = \mathsf{vk}(s')$.

R is a recipe such that $R\phi\downarrow = \mathsf{pub}(s)$ for some atom s.

- Case R is *not* in forced normal form, we have that $R\downarrow$ is a smaller recipe than R. By Lemma 3, $R\downarrow\phi\downarrow = R\phi\downarrow$. So by induction hypothesis $R\downarrow\psi\downarrow = \mathsf{pub}(s)$ for some atom s. We have already proved that, as $R\phi\downarrow$ is a message, $R\psi\downarrow$ is a message. So by Lemma 3, $R\downarrow\phi\downarrow = R\phi\downarrow = \mathsf{pub}(s)$.
- Case R is a simple recipe. In case R is a destructor-only recipe, we conclude relying on our hypothesis. Otherwise $R = \mathsf{pub}(R_1)$ and $R_1\phi\downarrow$ is an atom, thus R_1 is destructor-only. We conclude that $R_1\psi\downarrow$ is an atom too relying on our induction hypothesis, and thus $R\psi\downarrow = \mathsf{pub}(s')$ for some atom s'.

Second implication: $\phi \sqsubseteq_s^{\mathsf{simple}^+} \psi \Rightarrow \phi \sqsubseteq_s^{\mathsf{simple}} \psi$. Let R be a simple recipe with constructor context C and R' be a destructor-only recipe such that $R\phi\downarrow$ and $R'\phi\downarrow$ are both messages. Assume $R\phi\downarrow = R'\phi\downarrow$. We have to show that $R\psi\downarrow = R'\psi\downarrow$. We show that such a test transfers from ϕ to ψ by induction on $\#_{\mathsf{senc}}(R)$ (number of occurrences of senc and $\langle \dots \rangle_k$ at top level in C), and $\#_{\mathsf{adec}}(R')$ (number of occurrences of adec at top level in R') ordered lexicographically.

Base case: $(\#_{\mathsf{senc}}(R), \#_{\mathsf{adec}}(R')) = (0,0)$. In such a case, the test transfers by hypothesis.

Induction step: $\#_{\mathsf{senc}}(R) \geq 1$. In such a case, we consider $R = \mathsf{senc}(R_1, R_2)$. We consider the test $R_1 = \mathsf{sdec}(R', R_2)$. We have $(R_1 = \mathsf{sdec}(R', R_2))\phi$ so by induction hypothesis, $(R_1 = \mathsf{sdec}(R', R_2))\psi$. We deduce that $(R' = \mathsf{senc}(R_1, R_2))\psi$ that is $(R = R')\psi$. The case where $R = \langle R_1, R_2 \rangle$ can be done in a similar way.

Now, we consider the case where $\#_{\mathsf{senc}}(R) = 0$ and $\#_{\mathsf{adec}}(R') \geq 1$. In such a case, we have that $R' = \mathsf{adec}(R'_1, R'_2)$. We consider the test $\mathsf{aenc}(R, \mathsf{pub}(R'_2)) =$

R'_1. We still have that $\#_{\mathsf{senc}}(\mathsf{aenc}(R, \mathsf{pub}(R'_2))) = 0$, and $\#_{\mathsf{adec}}(R'_1) < \#_{\mathsf{adec}}(R')$. $(\mathsf{aenc}(R, \mathsf{pub}(R'_2)) = R'_1)\phi$ so by induction hypothesis $(\mathsf{aenc}(R, \mathsf{pub}(R'_2)) = R'_1)\psi$. We deduce that $(R = \mathsf{adec}(R'_1, R'_2))\psi$, that is $(R = R')\psi$. □

Let t be a term, we define $St_{\mathsf{ded}}(t)$ the set of subterms that occur at a deducible position as follows:

- $St_{\mathsf{ded}}(\langle t_1, \ldots, t_n \rangle_n) = \{\langle t_1, \ldots, t_n \rangle_n\} \cup St_{\mathsf{ded}}(t_1) \cup \ldots \cup St_{\mathsf{ded}}(t_n)$;
- $St_{\mathsf{ded}}(\mathsf{f}(t_1, t_2)) = \{\mathsf{f}(t_1, t_2)\} \cup St_{\mathsf{ded}}(t_1)$ with $\mathsf{f} \in \{\mathsf{senc}, \mathsf{aenc}, \mathsf{sign}\}$;
- $St_{\mathsf{ded}}(\mathsf{f}(t)) = \{\mathsf{f}(t)\}$ with $\mathsf{f} \in \{\mathsf{hash}, \mathsf{pub}, \mathsf{vk}\}$.

Lemma 2. *Let ϕ be a frame, $R = C[R_1, \ldots, R_k]$ be a simple recipe such that $\mathsf{root}(R) \notin \{\mathsf{senc}\} \cup \{\langle\ \rangle_k \mid 2 \leq k \leq n\}$, and R' be a destructor-only recipe such that $\mathsf{root}(R') \neq \mathsf{adec}$. Assume that $R\phi\!\downarrow$ and $R'\phi\!\downarrow$ are both messages such that $R\phi\!\downarrow = R'\phi\!\downarrow$. We have that either C is the empty context, or $R\phi\!\downarrow \in St_{\mathsf{opti}}(\phi) \cup \mathcal{C}_0$.*

Proof. Let ϕ be a frame, R' be a destructor-only recipe such that $R'\phi\!\downarrow$ is a message. We first prove that $R'\phi\!\downarrow \in St_{\mathsf{ded}}(\phi) \cup \mathcal{C}_0$ by structural induction on R'.

Base case $R' = \mathsf{w} \in dom(\phi)$ *or* $R' = a \in \mathcal{C}_0$. In both cases, we easily conclude that $R'\phi\!\downarrow \in St_{\mathsf{ded}}(\phi) \cup \mathcal{C}_0$.

Induction step: $R' = \mathsf{g}(R'_1, \ldots, R'_k)$. We distinguish several cases depending on g. In case $\mathsf{g} = \mathsf{sdec}$, we have that $k = 2$, $R'_1\phi\!\downarrow = \mathsf{senc}(t_1, t_2)$, $R'_2\phi\!\downarrow = t_2$, and $R'\phi\!\downarrow = t_1$ for some t_1, t_2. Applying our induction hypothesis on R'_1, we deduce that $R'_1\phi\!\downarrow \in St_{\mathsf{ded}}(\phi) \cup \mathcal{C}_0$, and thus $R'\phi\!\downarrow \in St_{\mathsf{ded}}(\phi) \cup \mathcal{C}_0$. The case where $R' = \langle R'_1, R'_2 \rangle$, $R' = \mathsf{getmsg}(R'_1)$, and $R' = \mathsf{adec}(R'_1, R'_2)$ are similar.

Now, we prove the proposition. Let $R = C[R_1, \ldots, R_k]$ be a simple recipe such that $\mathsf{root}(R) \notin \{\mathsf{senc}\} \cup \{\langle\ \rangle_k \mid 2 \leq k \leq n\}$, and R' be a destructor-only recipe such that $\mathsf{root}(R') \neq \mathsf{adec}$. Thanks to the result we have just proved, we know that $R'\phi\!\downarrow \in St_{\mathsf{ded}}(\phi) \cup \mathcal{C}_0$. Moreover, we know that either *(i)* $R' = \mathsf{w} \in dom(\phi)$ or *(ii)* $R' = a \in \mathcal{C}_0$ or *(iii)* $R' = \mathsf{g}(R'_1, \ldots, R'_k)$ for some $\mathsf{g} \in \Sigma_{\mathsf{d}} \setminus \{\mathsf{adec}\}$. In cases *(i)* and *(ii)*, we easily conclude that either C is empty or $R\phi\!\downarrow = C[R_1\phi\!\downarrow, \ldots, R_k\phi\!\downarrow] \in St_{\mathsf{opti}}(\phi) \cup \mathcal{C}_0$. In case *(iii)*, assuming that $\mathsf{g} = \mathsf{sdec}(R'_1, R'_2)$, we deduce that $R'_1\phi\!\downarrow \in St_{\mathsf{ded}}(\phi)$ and since $R'_1\phi\!\downarrow = \mathsf{senc}(t_1, t_2)$, we deduce that $R'_1\phi\!\downarrow \in St_{\mathsf{opti}}(\phi)$, and thus $R'\phi\!\downarrow \in St_{\mathsf{opti}}(\phi)$ since $R'\phi\!\downarrow = R\phi\!\downarrow$ is headed neither with a symbol in $\{\langle\ \rangle_k \mid 2 \leq k \leq n\}$, nor with senc. The case where $\mathsf{g} \in \{\langle\ \rangle_k \mid 2 \leq k \leq n\} \cup \{\mathsf{getmsg}\}$ can be done in a similar way. Note that we know that $\mathsf{g} \neq \mathsf{adec}$. □

Before establishing Proposition 1, we start by establishing the following result regarding deduction using destructor-only recipes.

Lemma 5. *Let ϕ and ψ be two frames having the same domain, and $\mathcal{C} \subseteq \mathcal{C}_0$. Let $\Theta = \langle \mathcal{F}act_0, \mathsf{Fact}_{\mathcal{C}}(\phi, \psi), \mathsf{Concrete}^{+}(\mathsf{R}_{\mathsf{Ana}}) \rangle$ and $\Pi = \langle \Theta, \{\mathsf{att}(u, v)\} \rangle$ for some messages u and v. If Π has a solution then there is a destructor-only $\mathcal{C}$-recipe R such that $R\phi\!\downarrow = u$, and $R\psi\!\downarrow = v$.*

Proof. Let $\pi = r_1, \ldots, r_n$ be a planning path from S_0 to S_n, and $\mathsf{att}(u, v) \in S_n$. We show the result by induction on the length of π, and since applying planning rules in parallel does not change the set of reachable states, we assume w.l.o.g. that the planning path under study is such that each $r_i \in \mathsf{Concrete}^+(\mathsf{R}_{\mathsf{Ana}})$.

Base case. We have that π is empty. In such a case, by definition of S_0, the result trivially holds.

Inductive case. We know that r_n is an instance of one of the abstract rules in $\mathsf{R}_{\mathsf{Ana}}$, *e.g.* $\mathsf{att}(\mathsf{senc}(u_1, u_2), \mathsf{senc}(v_1, v_2)), \mathsf{att}(u_2, v_2) \to \mathsf{att}(u_1, v_1)$. Thanks to our induction hypothesis, we know that there exist:

- a destructor-only $\mathcal{C}$-recipe R_1 such that $R_1\phi{\downarrow} = \mathsf{senc}(u_1, u_2)$, and $R_1\psi{\downarrow} = \mathsf{senc}(v_1, v_2)$;
- a destructor-only $\mathcal{C}$-recipe R_2 such that $R_2\phi{\downarrow} = u_2$, and $R_2\psi{\downarrow} = v_2$.

Therefore, the $\mathcal{C}$-recipe $R = \mathsf{sdec}(R_1, R_2)$ allows us to conclude. The reasoning is rather similar for the other rule. □

Let R be a destructor-only recipe. We denote by $|R|_{\mathsf{main}}$ the length of its *main path*, i.e. the longest sequence of 1 which corresponds to a position in R.

Lemma 6. *Let ϕ be a frame, and R be a destructor-only recipe such that $R\phi{\downarrow}$ is a message. We have that $|R|_{\mathsf{main}} \leq \mathsf{depth}(\phi)$.*

Proof. We first establish that $|R|_{\mathsf{main}} + \mathsf{depth}(R\phi{\downarrow}) \leq \mathsf{depth}(\phi)$ for any R such that $R\phi{\downarrow}$ is a message by structural induction on R.

Base case: $R = \mathsf{w}$ or $R \in \mathcal{C}_0$. In such a case, we have

$$|R|_{\mathsf{main}} + \mathsf{depth}(R\phi{\downarrow}) = 0 + \mathsf{depth}(R\phi{\downarrow}) \leq \mathsf{depth}(\phi).$$

Inductive case: $R = \mathsf{g}(R_1, \ldots, R_k)$ for some $\mathsf{g} \in \Sigma_{\mathsf{d}}$. Since $R\phi{\downarrow}$ is a message, we know that $R_1\phi{\downarrow}$ is a message. Therefore, the induction hypothesis applies and we obtain that $|R_1|_{\mathsf{main}} + \mathsf{depth}(R_1\phi{\downarrow}) \leq \mathsf{depth}(\phi)$. Moreover, we have that $\mathsf{depth}(R_1\phi{\downarrow}) \geq 1 + \mathsf{depth}(R\phi{\downarrow})$, and $|R|_{\mathsf{main}} = |R_1|_{\mathsf{main}} + 1$. Therefore, we have that:

$$|R|_{\mathsf{main}} + \mathsf{depth}(R\phi{\downarrow}) \leq |R_1|_{\mathsf{main}} + 1 + \mathsf{depth}(R_1\phi{\downarrow}) - 1 \leq \mathsf{depth}(\phi).$$

We have that $|R|_{\mathsf{main}} + \mathsf{depth}(R\phi{\downarrow}) \leq \mathsf{depth}(\phi)$. Since $\mathsf{depth}(R\phi{\downarrow}) \geq 0$, we deduce that $|R|_{\mathsf{main}} \leq \mathsf{depth}(\phi)$. □

Given a destructor-only recipe R, the *key-recipes* of R, denoted $\mathsf{Key}(R)$, are all the recipes occurring as a subterm of R at a key position, i.e. as a second argument of sdec or adec, excepted those of the form $c \in \mathcal{C}_0$ that do not count. This notation is extended as expected to sets of destructor-only recipes. Given a set of recipes $\mathcal{R}$, we write $|\mathcal{R}|$ the number of elements occurring in it.

Lemma 7. *Let ϕ and ψ be two frames having the same domain, and $\mathcal{R}$ be a set of destructor-only $\mathcal{C}$-recipes such that $R\phi{\downarrow}$ and $R\psi{\downarrow}$ are messages for any $R \in \mathcal{R}$. Let $\Theta = \langle \mathcal{F}act_0, \mathsf{Fact}_{\mathcal{C}}(\phi, \psi) \cup \{\mathsf{att}(R\phi{\downarrow}, R\psi{\downarrow}) \mid R \in \mathsf{Key}(\mathcal{R})\}, \mathsf{Concrete}^+(\mathsf{R}_{\mathsf{Ana}})\rangle$, and $\Pi = \langle \Theta, \{\mathsf{att}(R\phi{\downarrow}, R\psi{\downarrow}) \mid R \in \mathcal{R}\}\rangle$. We have that Π has a solution of length at most $\mathsf{depth}(\phi)$.*

Proof. We show the existence of a solution of length at most $\ell = \mathsf{max}\{|R|_{\mathsf{main}} \mid R \in \mathcal{R}\}$, and we conclude relying on Lemma 6 that $\ell \leq \mathsf{depth}(\phi)$.

Base case: $\ell = 0$. In such a case, we have that $R \in dom(\phi) \cup \mathcal{C}$ for any $R \in \mathcal{R}$, and the empty planning path of length 0 is indeed a solution of Π.

Inductive case: $\ell > 0$. Let $\mathcal{R}' = \{R \in \mathcal{R} \mid |R|_{\mathsf{main}} = \ell\} = \{R_1, \ldots, R_n\}$. For each $i \in \{1, \ldots, n\}$, we have that:

- either $R_i = \mathsf{des}(R_i^1, R_i^2)$ with $\mathsf{des} \in \{\mathsf{adec}, \mathsf{sdec}\}$; and $R_i^1\phi{\downarrow}$ and $R_i^1\psi{\downarrow}$, as well as $R_i^2\psi{\downarrow}$ and $R_i^2\psi{\downarrow}$ are messages. Moreover, we have that $R_i^2 \in \mathsf{Key}(R_i)$ or $R_i^2 \in \mathcal{C}_0$, and thus by hypothesis, we have that $\mathsf{att}(R_i^2\phi{\downarrow}, R_i^2\psi{\downarrow})$ belongs to the initial state of our planning system Θ.
- or $R_i = \mathsf{des}(R_i^1)$ with $\mathsf{des} \in \{\mathsf{proj}_\ell^k, \mathsf{getmsg}\}$; and $R_i^1\phi{\downarrow}$ and $R_i^1\psi{\downarrow}$ are messages.

Moreover, we have that $\mathsf{Key}(R_i^1) \subseteq \mathsf{Key}(R')$ for any $i \in \{1, \ldots, n\}$. Therefore, relying on our induction hypothesis, we deduce that there is a plan π_0 of length $\ell - 1$ of

$$\Pi_0 = \langle \Theta, \{\mathsf{att}(R_i^1\phi{\downarrow}, R_i^1\psi{\downarrow}) \mid 1 \leq i \leq n\} \cup \mathsf{att}(R'\phi{\downarrow}, R'\psi{\downarrow}) \mid R' \in \mathcal{R} \setminus \mathcal{R}'\}\rangle.$$

Moreover, when $R_i = \mathsf{sdec}(R_i^1, R_i^2)$ (the case where $R_i = \mathsf{adec}(R_i^1, R_i^2)$ can be done in a similar way), as $R_i\phi{\downarrow}$ and $R_i\psi{\downarrow}$ are messages, we know that $R_i^1\phi{\downarrow} = \mathsf{senc}(R_i\phi{\downarrow}, R_i^2\phi{\downarrow})$ and $R_i^1\psi{\downarrow} = \mathsf{senc}(R_i\psi{\downarrow}, R_i^2\psi{\downarrow})$. Let r_i be the following planning rule:

$$\mathsf{att}(\mathsf{senc}(R_i\phi{\downarrow}, R_i^2\phi{\downarrow}), \mathsf{senc}(R_i\psi{\downarrow}, R_i^2\psi{\downarrow})),\ \mathsf{att}(R_i^2\phi{\downarrow}, R_i^2\psi{\downarrow}) \rightarrow \mathsf{att}(R_i\phi{\downarrow}, R_i\psi{\downarrow})$$

We have that:

- $\mathsf{Add}(r_i) = \{\mathsf{att}(R_i\phi{\downarrow}, R_i\psi{\downarrow})\}$;
- $\mathsf{att}(\mathsf{senc}(R_i\phi{\downarrow}, R_i^2\phi{\downarrow}), \mathsf{senc}(R_i\psi{\downarrow}, R_i^2\psi{\downarrow})) = \mathsf{att}(R_i^1\phi{\downarrow}, R_i^1\psi{\downarrow})$, and
- $\mathsf{att}(R_i^2\phi{\downarrow}, R_i^2\psi{\downarrow})$ belongs to the initial state of our planning system Θ.

So each r_i $(1 \leq i \leq n)$ is applicable after π_0, and therefore $\pi_0.\{r_1, \ldots, r_n\}$ is a planning path of length ℓ which is a solution of Π. □

Lemma 8. *Let ϕ and ψ be two frames having the same domain, and $\mathcal{R}$ be a set of destructor-only $\mathcal{C}$-recipes such that $\mathsf{Key}(\mathcal{R}) \subseteq \mathcal{R}$; $R\phi{\downarrow}$ and $R\psi{\downarrow}$ are messages for any $R \in \mathcal{R}$.*
Let $\hat{\mathcal{F}} = \{\mathsf{att}(R\phi{\downarrow}, R\psi{\downarrow}) \mid R \in \mathcal{R}\}$, $\mathcal{R}_0 \subseteq \mathcal{R}$, and $\hat{\mathcal{F}}_0 = \{\mathsf{att}(R\phi{\downarrow}, R\psi{\downarrow}) \mid R \in \mathcal{R}_0\}$.
Let $\Theta = \langle \mathcal{F}act_0, \mathsf{Fact}_{\mathcal{C}}(\phi, \psi) \cup \hat{\mathcal{F}}_0, \mathsf{Concrete}^+(\mathsf{Rule}_A)\rangle$, and $\Pi = \langle \Theta, \hat{\mathcal{F}}\rangle$. We have that Π has a solution of length at most $|\mathcal{R} \setminus \mathcal{R}_0| \times \mathsf{depth}(\phi)$.

Proof. We consider the partial order $R < R'$ over destructor-only recipes such that $R < R'$ if, and only if, $R \in \mathsf{Key}(R')$. We show the result by induction on $|\mathcal{R} \setminus \mathcal{R}_0|$, i.e. the number of elements in $\mathcal{R} \setminus \mathcal{R}_0$.

Base case: $|\mathcal{R} \setminus \mathcal{R}_0| = 0$. In such a case, the empty plan of length 0 is indeed a solution of $\Pi = \langle \Theta, \mathcal{F}\rangle$.

Inductive case: $|\mathcal{R} \setminus \mathcal{R}_0| > 0$. In such a case, we consider R a minimal element in $\mathcal{R} \setminus \mathcal{R}_0$ w.r.t. $<$. We have that $R\phi\downarrow$ and $R\psi\downarrow$ are messages. We know that $\mathsf{Key}(R) \subseteq \mathcal{R}$, and since R is minimal, we have that $\mathsf{Key}(R) \subseteq \mathcal{R}_0$. Therefore, Lemma 7 applies to $\{R\}$, and we get a solution π of length at most $\mathsf{depth}(\phi)$ of $\langle \Theta, \{\mathsf{att}(R\phi\downarrow, R\psi\downarrow)\}\rangle$.

Let $\Theta' = \langle \mathcal{F}_0, \mathsf{Fact}_{\mathcal{C}}(\phi, \psi) \cup \hat{\mathcal{F}}_0 \cup \mathsf{att}(R\phi\downarrow, R\psi\downarrow), \mathsf{Concrete}^+(\mathsf{Rule}_A)\rangle$. By induction hypothesis, there is a solution π' of $\langle \Theta', \mathcal{F}\rangle$ of length at most $|\mathcal{R} \setminus (\mathcal{R}_0 \cup \{R\})| \times \mathsf{depth}(\phi) = |\mathcal{R} \setminus \mathcal{R}_0| \times \mathsf{depth}(\phi) - \mathsf{depth}(\phi)$. We conclude that $\pi.\pi'$ is a solution of Π of length at most $|\mathcal{R} \setminus \mathcal{R}_0| \times \mathsf{depth}(\phi)$. □

A static inclusion test T w.r.t. $\sqsubseteq_s^{\mathsf{simple}}$ is either a destructor-only recipe R, or a pair (R, R') of recipes such that R is simple whereas R' is destructor-only. We define $\mathsf{Key}(T)$ as:

- $\mathsf{Key}(R)$ in the first case,
- $\mathsf{Key}(\{R', R_1, \ldots, R_n\})$ in the second case assuming that $R = C[R_1, \ldots, R_n]$.

Lemma 9. *Let ϕ and ψ be two frames with $dom(\phi) = dom(\psi)$. Let $\mathcal{R}_0$ be a set of destructor-only $\mathcal{C}_0$-recipes such that $\mathsf{Key}(\mathcal{R}_0) \subseteq \mathcal{R}_0$; $R\phi\downarrow$ and $R\psi\downarrow$ are messages for any $R \in \mathcal{R}_0$. Let $\hat{\mathcal{F}} = \{\mathsf{att}(R\phi\downarrow, R\psi\downarrow) \mid R \in \mathcal{R}_0\}$ and $\Theta = \langle \mathcal{F}act_0, \mathsf{Fact}_{\mathcal{C}_0}(\phi, \psi) \cup \hat{\mathcal{F}}, \mathcal{R}\rangle$ where*

$$\mathcal{R} = \mathsf{Concrete}(\mathsf{R}_{\mathsf{Ana}}) \cup \mathcal{R}_{\mathsf{fail}}^{\mathsf{test}_1} \cup \mathcal{R}_{\mathsf{fail}}^{\mathsf{test}_2} \cup \mathcal{R}_{\mathsf{fail}}^{\mathsf{atom}} \cup \mathcal{R}_{\mathsf{fail}}^{\mathsf{check}} \cup \mathcal{R}_{\mathsf{fail}}^{\mathsf{pub}}.$$

Let $\Pi = \langle \Theta, \{\mathsf{bad}\}\rangle$. Assume $\phi \not\sqsubseteq_s^{\mathsf{simple}^+} \psi$. Let T be a static inclusion test w.r.t. $\sqsubseteq_s^{\mathsf{simple}}$ that holds in ϕ but not in ψ. Then Π has a solution of length at most $(k+1) \times \mathsf{depth}(\phi) + 1$ where $k = |\mathsf{Key}(T) \setminus \mathcal{R}_0|$.

Proof. We have that $\phi \not\sqsubseteq_s^{\mathsf{simple}^+} \psi$. Following the definition of $\sqsubseteq_s^{\mathsf{simple}^+}$, we consider the four cases separately.

1. There is a destructor-only recipe R such that $R\phi\downarrow$ is a message but $R\psi\downarrow$ is not. Let R' be the smallest subterm of R such that this property holds. Since both ϕ and ψ are frames, and thus contain messages, we have that $R' = \mathsf{g}(R'_1, R'_2)$ with $\mathsf{g} \in \{\mathsf{sdec}, \mathsf{adec}\}$, or $R' = \mathsf{g}(R'_1)$ with $\mathsf{g} \in \{\mathsf{getmsg}\} \cup \{\mathsf{proj}_j^k \mid 2 \leq k \leq n, 1 \leq j \leq k\}$. We assume w.l.o.g. that $R' = \mathsf{sdec}(R'_1, R'_2)$, and by minimality of our test, $R'_1\psi\downarrow$ and $R'_2\psi\downarrow$ are messages.
   Applying Lemma 8 with $\mathcal{R}' = \mathsf{Key}(\{R'_1, R'_2\}) \cup \{R'_2\}$ and $\mathcal{R}_0$, we deduce the existence of a plan of length at most $|\mathsf{Key}(\{R'_1, R'_2\}) \cup \{R'_2\} \setminus \mathcal{R}_0| \times \mathsf{depth}(\phi)$ leading to
   $$\{\mathsf{att}(R\phi\downarrow, R\psi\downarrow) \mid R \in \mathsf{Key}(\{R'_1, R'_2\}) \cup \{R'_2\}\}.$$
   Thus, relying on Lemma 7, $\langle \Theta, \{\mathsf{att}(R'_1\phi\downarrow, R'_1\psi\downarrow), \mathsf{att}(R'_2\phi\downarrow, R'_2\psi\downarrow)\}\rangle$ has a solution of length at most
   $$|(\mathsf{Key}(\{R'_1, R'_2\}) \cup \{R'_2\}) \setminus \mathcal{R}_0| \times \mathsf{depth}(\phi) + \mathsf{depth}(\phi).$$
   Then, we consider the rule r of the form:
   $$\mathsf{att}(R'_1\phi\downarrow, R'_1\psi\downarrow), \mathsf{att}(R'_2\phi\downarrow, R'_2\psi\downarrow) \to \mathsf{bad}$$

which is indeed an instance of a rule in $\mathsf{Concrete}^{-}(\mathsf{R}_{\mathsf{Ana}})$ since $R'_1\phi{\downarrow} = \mathsf{senc}(u_1, u_2)$ for some u_1, u_2, and $\mathsf{senc}(u_1, u_2)$, $R'_1\psi{\downarrow}$, $R'_2\psi{\downarrow}$ are messages whereas $\mathsf{sdec}(R'_1\psi{\downarrow}, R'_2\psi{\downarrow}){\downarrow}$ is not a message. This rule r can be triggered and leads to bad. Since $\mathsf{Key}(\{R'_1, R'_2\} \cup \{R'_2\}) \subseteq \mathsf{Key}(R') \subseteq \mathsf{Key}(R) = \mathsf{Key}(T)$, we have that Π has as solution of length at most

$$(|\mathsf{Key}(T) \setminus \mathcal{R}_0| + 1) \times \mathsf{depth}(\phi) + 1.$$

Now, assuming that $R\phi{\downarrow}$ is an atomic message, and $R\psi{\downarrow}$ is a message but not an atomic one. In such a case, with a similar reasoning, we get that $\langle \Theta, \{\mathsf{att}(R\phi{\downarrow}, R\psi{\downarrow})\}\rangle$ has a solution of length at most

$$|\mathsf{Key}(T) \setminus \mathcal{R}_0| \times \mathsf{depth}(\phi) + \mathsf{depth}(\phi).$$

Then, we consider the rule r of the form:

$$\mathsf{att}(R\phi{\downarrow}, R\psi{\downarrow}) \to \mathsf{bad}$$

which is indeed an instance of a rule $\mathcal{R}^{\mathsf{atom}}_{\mathsf{fail}}$, and which leads to a solution for Π. Therefore, Π has as solution of length at most

$$(|\mathsf{Key}(T) \setminus \mathcal{R}_0| + 1) \times \mathsf{depth}(\phi) + 1.$$

2. The minimal test is a destructor-only recipe R such that $R\phi{\downarrow} = \mathsf{pub}(s)$ for some atom s, whereas $R\psi{\downarrow} \neq \mathsf{pub}(s')$ for any atom s'. First, thanks to the first item, we may assume that $R\psi{\downarrow}$ is a message. Applying Lemma 8 with $\mathcal{R} = \mathsf{Key}(R)$ and $\mathcal{R}_0$, and then Lemma 7 on $\{R\}$, we get a solution of $\langle \Theta, \{\mathsf{att}(R\phi{\downarrow}, R\psi{\downarrow})\}\rangle$ of length at most $(|\mathsf{Key}(R)| + 1) \times \mathsf{depth}(\phi)$. Then, we consider the rule r of the form:

$$\mathsf{att}(R\phi{\downarrow}, R\psi{\downarrow}) \to \mathsf{bad}$$

which is indeed an instance of a rule $\mathcal{R}^{\mathsf{pub}}_{\mathsf{fail}}$, and which leads to a solution for Π of length at most $(|\mathsf{Key}(T) \setminus \mathcal{R}_0| + 1) \times \mathsf{depth}(\phi) + 1$.

3. There are destructor-only recipes R and R' such that $R\phi{\downarrow} = \mathsf{sign}(t, s)$, $R'\phi{\downarrow} = \mathsf{vk}(s)$ for some message t, and some atom s, whereas $R\psi{\downarrow} \neq \mathsf{sign}(t', s')$ or $R\psi{\downarrow} \neq \mathsf{vk}(s')$ for any message t' and any atom s'. First, thanks to the first item, we may assume that both $R\psi{\downarrow}$, and $R'\psi{\downarrow}$ are messages. Applying Lemma 8 with $\mathcal{R} = \mathsf{Key}(\{R, R'\})$ and $\mathcal{R}_0$, and then Lemma 7 on $\{R, R'\}$, we get a solution of $\langle \Theta, \{\mathsf{att}(R\phi{\downarrow}, R\psi{\downarrow}), \mathsf{att}(R'\phi{\downarrow}, R'\psi{\downarrow})\}\rangle$ of length at most $(|\mathsf{Key}(\{R, R'\}) \setminus \mathcal{R}_0| + 1) \times \mathsf{depth}(\phi)$. Then, we consider the rule r of the form:

$$\mathsf{att}(R\phi{\downarrow}, R\psi{\downarrow}), \mathsf{att}(R'\phi{\downarrow}, R'\psi{\downarrow}) \to \mathsf{bad}$$

which is indeed an instance of a rule $\mathcal{R}^{\mathsf{check}}_{\mathsf{fail}}$, and which leads to a solution for Π of length at most $(|\mathsf{Key}(T) \setminus \mathcal{R}_0| + 1) \times \mathsf{depth}(\phi) + 1$.

4. There is a simple recipe $R = C[R_1, \ldots, R_i]$ and a destructor-only recipe R' such that $R\phi{\downarrow} = R'\phi{\downarrow}$ are messages whereas $R\psi{\downarrow} \neq R'\psi{\downarrow}$. Moreover, we

know that $\mathsf{root}(R) \notin \{\mathsf{senc}\} \cup \{\langle\,\rangle_k \mid 2 \leq k \leq n\}$, and also that $\mathsf{root}(R') \neq \mathsf{adec}$. First, thanks to the first item, we may assume that $R_1\psi\!\downarrow, \ldots, R_k\psi\!\downarrow$, as well as $R\psi\!\downarrow$, are messages. Applying Lemma 8 on $\mathsf{Key}(\{R_1, \ldots, R_i, R'\})$, and then Lemma 7 on $\{R_1, \ldots, R_i, R'\}$, we get a solution of

$$\langle \Theta, \{\mathsf{att}(R_1\phi\!\downarrow, R_1\psi\!\downarrow), \ldots, \mathsf{att}(R_i\phi\!\downarrow, R_i\psi\!\downarrow), \mathsf{att}(R'\phi\!\downarrow, R'\psi\!\downarrow)\}\rangle$$

of length at most $(|\mathsf{Key}(\{R_1, \ldots, R_i, R'\}) \setminus \mathcal{R}_0| + 1) \times \mathsf{depth}(\phi)$. Then, we consider the rule r of the form:

$$\mathsf{att}(R_1\phi\!\downarrow, R_1\psi\!\downarrow), \ldots, \mathsf{att}(R_i\phi\!\downarrow, R_i\psi\!\downarrow), \mathsf{att}(R'\phi\!\downarrow, R'\psi\!\downarrow) \to \mathsf{bad}$$

which is an instance of a rule in $\mathcal{R}_{\mathsf{fail}}^{\mathsf{test}_1}$ or $\mathcal{R}_{\mathsf{fail}}^{\mathsf{test}_2}$. Indeed, thanks to Lemma 2, we know that:

- either C is the empty context, and thus r is an instance of $\mathcal{R}_{\mathsf{fail}}^{\mathsf{test}_1}$ since $R'\phi\!\downarrow = R\phi\!\downarrow$, and $R'\psi\!\downarrow \neq R\psi\!\downarrow$;
- or $R\phi\!\downarrow \in St_{\mathsf{opti}}(\phi) \cup \mathcal{C}_0$, and thus r is an instance of $\mathcal{R}_{\mathsf{fail}}^{\mathsf{test}_2}$ since $R'\phi\!\downarrow = R\phi\!\downarrow = C[R_1\phi\!\downarrow, \ldots, R_i\phi\!\downarrow]$, and $R'\psi\!\downarrow \neq R\psi\!\downarrow = C[R_1\psi\!\downarrow, \ldots, R_i\psi\!\downarrow]$.

Therefore, this allows us to conclude that Π has a solution of length at most $(|\mathsf{Key}(T) \setminus \mathcal{R}_0| + 1) \times \mathsf{depth}(\phi) + 1$. □

Proposition 1. *Let ϕ and ψ be two frames with $dom(\phi) = dom(\psi)$, and $\Theta = \langle \mathcal{F}act_0, \mathsf{Fact}_{\mathcal{C}_0}(\phi, \psi), \mathcal{R}\rangle$ where*

$$\mathcal{R} = \mathsf{Concrete}(\mathsf{R}_{\mathsf{Ana}}) \cup \mathcal{R}_{\mathsf{fail}}^{\mathsf{test}_1} \cup \mathcal{R}_{\mathsf{fail}}^{\mathsf{test}_2} \cup \mathcal{R}_{\mathsf{fail}}^{\mathsf{atom}} \cup \mathcal{R}_{\mathsf{fail}}^{\mathsf{check}} \cup \mathcal{R}_{\mathsf{fail}}^{\mathsf{pub}}.$$

Let $\Pi = \langle \Theta, \{\mathsf{bad}\}\rangle$. We have that $\phi \not\sqsubseteq_s \psi$ if, and only if, Π has a solution of length at most $(N+1) \times \mathsf{depth}(\phi) + 1$ where N is the number of names n occurring in ϕ at a key position, i.e. such that n (resp. $\mathsf{pub}(n)$) occurs in key position of an encryption in ϕ.

Proof. First, thanks to Lemma 1, it is sufficient to show that $\phi \not\sqsubseteq_s^{\mathsf{simple}^+} \psi$ if, and only if, Π has a solution. We show the two directions separately.

($\Rightarrow$) We have that $\phi \not\sqsubseteq_s^{\mathsf{simple}^+} \psi$. We consider a static inclusion test T w.r.t. $\sqsubseteq_s^{\mathsf{simple}^+}$. We consider one such that $\mathsf{Key}(T)$ is minimal, and then among those that have a minimal $\mathsf{Key}(T)$, we consider one whose size (number of function symbols) is minimal. Lemma 9 applies with $\mathcal{R}_0 = \emptyset$. Therefore, we have that Π has a solution of length at most $(|\mathsf{Key}(T)|+1)\times\mathsf{depth}(\phi)+1$. To conclude, we need to show that $|\mathsf{Key}(T)| \leq N$. Actually, we have that $\{R\phi\!\downarrow \mid R \in \mathsf{Key}(T)\} \setminus \mathcal{C}_0 \leq N$ since all these recipes occur in key position (i.e. second argument of $\mathsf{sdec}/\mathsf{adec}$) of a recipe that leads to a message in ϕ. Now, in case there are two recipes in $K, K' \in \mathsf{Key}(T) \cup \mathcal{C}_0$ such that $K\phi\!\downarrow = K'\phi\!\downarrow$, then by minimality of our test (note that at least the size of the test has descreased), we know that $K\psi\!\downarrow = K'\psi\!\downarrow$, and therefore replacing K by K' will give us a smaller test according to our measure. This allows us to conclude that Π has a solution of length at most $(N+1) \times \mathsf{depth}(\phi) + 1$.

($\Leftarrow$) We have a plan of bad. We consider such a plan $r_1, \ldots, r_n$ of minimal length without performing rules in parallel. Since this plan is mimimal, we know

that $r_1, \ldots, r_{n-1}$ are rules in $\mathsf{Concrete}^+(\mathsf{R}_{\mathsf{Ana}})$, and therefore, we can rely on Lemma 5 to conclude that there exists a destructor-only recipe R such that $R\phi{\downarrow} = u$ and $R\psi{\downarrow} = v$ for any $\mathsf{att}(u, v) \in S_{n-1}$ (the state resulting from the application of $r_1, \ldots, r_{n-1}$). Then, in order to derive bad, we have applied a rule in $\mathsf{Concrete}^-(\mathsf{R}_{\mathsf{Ana}}) \cup \mathsf{Rule}_{\mathsf{fail}}^{\mathsf{test}_1} \cup \mathsf{Rule}_{\mathsf{fail}}^{\mathsf{test}_2} \cup \mathsf{Rule}_{\mathsf{fail}}^{\mathsf{atom}} \cup \mathsf{Rule}_{\mathsf{fail}}^{\mathsf{check}} \cup \mathsf{Rule}_{\mathsf{fail}}^{\mathsf{pub}}$. It is actually easy to derive a witness of $\phi \not\sqsubseteq_s \psi$. For instance, regarding the first case, according to the definition of $\mathsf{Concrete}^-(\mathsf{R}_{\mathsf{Ana}})$, we have for instance $r_n = \mathsf{att}(\mathsf{senc}(u_1, u_2), v), \mathsf{att}(u_2, v') \to \mathsf{bad}$ with v not of the form $\mathsf{senc}(v_0, v')$ for some v_0. Moreover, we have destructor-only recipes R_1 (and R_2) allowing us to derive these facts. In such a case, we conclude using the recipe $\mathsf{sdec}(R_1, R_2)$. The other cases are rather similar. □

B Reduction to well-typed witness

In this section, we exploit a result established in [17]. The main idea is to show that we can restrict ourselves to consider well-typed execution when looking for a witness of non0-inclusion. However, to achieve this, we need to assume the existence of an infinite set $\mathcal{C}_0^{\mathsf{bitstring}}$ of constants of sort $\mathsf{bitstring}$. In the following, we will consider $\mathcal{C}_0^+ = \mathcal{C}_0 \uplus \mathcal{C}_0^{\mathsf{bitstring}}$, and we further assume the existence of an infinite number of constants of any type in both sets.

All our definitions (messages, recipes, frames, configurations, processes, etc), as well as our semantics can be easily extended to take into account this new set of constants, and we sometimes parametrized these definition explicitly with the underlying set of constants (typically $\mathcal{C}_0$ or $\mathcal{C}_0^{\mathsf{bitstring}}$) to avoid confusion. Note that any constant in $\mathcal{C}_0^{\mathsf{bitstring}}$ is of sort $\mathsf{bitstring}$, and therefore not an atomic constant. Such a constant c can not occur at a key position, e.g. in $\mathsf{senc}(m, \mathsf{c})$.

Theorem 3. *Let $\mathcal{K}_\mathcal{P}$ be a $\mathcal{C}_0$-configuration type-compliant w.r.t. $(\mathcal{T}_\mathcal{P}, \delta_\mathcal{P})$ and $\mathcal{K}_\mathcal{Q}$ be a $\mathcal{C}_0$-configuration. We have that $\mathcal{K}_\mathcal{P} \not\sqsubseteq_t \mathcal{K}_\mathcal{Q}$ w.r.t. $\mathcal{C}_0$ if, and only if, there exists a witness $(\mathsf{tr}, \phi) \in \mathsf{trace}_{\mathcal{C}_0}(\mathcal{K}_\mathcal{P})$ of this non-inclusion which only involves simple recipes, and with a well-typed underlying execution.*

Theorem 3 is a direct consequence of the result established in [17]. It only remains to establish that the witness tr can be choosen such that it only involves simple recipes. Actually, this can be obtained by taking recipes in forced normal form, and it is a direct consequence of Lemma 3 and Lemma 4 (see Appendix A).

C Bounding constants

Our goal is to bound the number of constants used in an attack. We show here that actually, two constants are sufficient. The proof technique is inspired from [15] and [19] which respectively reduce the number of nonces and agents in the context of equivalence properties. A direct application of the proof technique would however yield two constants of each type, which represents still a high number of constants. Instead, we show here how to further reduce the number

of additional constants to only 3, considering a weaker version of well-typed execution, namely quasi-well-typed execution.

Given a protocol $\mathcal{P}$, we denote $\mathcal{C}_\mathcal{P}$ the constants from $\mathcal{C}_0$ that occur in $\mathcal{P}$. Sometimes, to make clear the set of constants $\mathcal{C}$ that may be used, we write $\to$ w.r.t. $\mathcal{C}$, $\mathsf{trace}_\mathcal{C}(\mathcal{K})$, $\mathcal{C}$-recipe, $\mathcal{C}$-message, ...

Theorem 1. *Let $\mathcal{K}_\mathcal{P}$ be an initial $\mathcal{C}_0$-configuration type-compliant w.r.t. $(\mathcal{T}_\mathcal{P}, \delta_\mathcal{P})$ and $\mathcal{K}_\mathcal{Q}$ be another initial $\mathcal{C}_0$-configuration. Let $\mathcal{C}^\star = (\mathcal{C}_\mathcal{P} \cup \mathcal{C}_\mathcal{Q}) \uplus \{c_0^\star, c_1^\star, c_+^\star\}$. We have that $\mathcal{K}_\mathcal{P} \not\sqsubseteq_t \mathcal{K}_\mathcal{Q}$ w.r.t. $\mathcal{C}_0$ if, and only if, there exists a witness $(\mathsf{tr}, \phi) \in \mathsf{trace}(\mathcal{K}_\mathcal{P})$ of this non-inclusion which only involves constants from $\mathcal{C}^\star$, simple recipes, and with a quasi-well-typed underlying execution.*

Given $\mathsf{A} \subseteq \mathcal{C}_0^+$, an A-renaming is a function ρ such that $dom(\rho) \cup img(\rho) \subseteq \mathsf{A}$; and $\rho(a)$ is of sort atom when a is of sort atom. Given a typing system $(\mathcal{T}, \delta)$, we say that ρ is *type-preserving* when $\delta(a) = \delta(\rho(a))$ for any $a \in dom(\rho)$.

Lemma 10. *Let t and t' be two terms in $\mathcal{T}(\Sigma, \mathcal{C}_0^+ \uplus \mathcal{N})$.*

1. *If $t{\downarrow}$ is a $\mathcal{C}_0^+$-message then $(t{\downarrow})\rho = (t\rho){\downarrow}$ for any $\mathcal{C}_0^+$-renaming ρ.*
2. *If $t{\downarrow}$ is not a $\mathcal{C}_0^+$-message, then there exists $c_0 \in \mathcal{C}_0^+$ such that for any $\mathcal{C}_0^+$-renaming ρ such that $c_0 \notin dom(\rho) \cup img(\rho)$, $t\rho{\downarrow}$ is not a $\mathcal{C}_0^+$-message.*
3. *If $t{\downarrow}$ and $t'{\downarrow}$ are $\mathcal{C}_0^+$-messages and $t{\downarrow} \neq t'{\downarrow}$, then there exists $c_0 \in \mathcal{C}_0^+$ such that for any $\mathcal{C}_0^+$-renaming ρ such that $c_0 \notin dom(\rho) \cup img(\rho)$, $t\rho{\downarrow} \neq t'\rho{\downarrow}$.*

Proof. We prove the three items separately.

Item 1. Let $t \in \mathcal{T}(\Sigma, \mathcal{C}_0^+ \uplus \mathcal{N})$ such that $t{\downarrow}$ is a $\mathcal{C}_0^+$-message. We show the result by structural induction on t.
Base case. In such a case, we have that $t \in \mathcal{C}_0^+ \uplus \mathcal{N}$, and thus $(t{\downarrow})\rho = t\rho = (t\rho){\downarrow}$.
Inductive case. First, we consider the case where $t = \mathsf{f}(t_1, \ldots, t_k)$ with $\mathsf{f} \in \Sigma_\mathsf{c}$. In such a case, relying on our induction hypothesis, we have that:

$$(t{\downarrow})\rho = \mathsf{f}(t_1{\downarrow}\rho, \ldots, t_k{\downarrow}\rho) = \mathsf{f}(t_1\rho{\downarrow}, \ldots, t_k\rho{\downarrow}) = \mathsf{f}(t_1, \ldots, t_k)\rho{\downarrow} = (t\rho){\downarrow}.$$

Second, we consider the case where $t = \mathsf{g}(t_1, \ldots, t_k)$ with $\mathsf{g} \in \Sigma_\mathsf{d} \cup \{\mathsf{check}\}$. For sake of clarity, we consider the case where $\mathsf{g} = \mathsf{adec}$. In such a case, we have that $t = \mathsf{adec}(t_1, t_2)$, $t_1{\downarrow}, t_2{\downarrow}$ are $\mathcal{C}_0^+$-messages, $t_1{\downarrow} = \mathsf{aenc}(u, \mathsf{pub}(v))$, $t_2{\downarrow} = v$ and $\mathsf{sdec}(t_1, t_2){\downarrow} = u$ for some $\mathcal{C}_0^+$-message u, and some atom v. Thanks to our induction hypothesis, we have that:

- $(t_1{\downarrow})\rho = \mathsf{aenc}(u, \mathsf{pub}(v))\rho = \mathsf{aenc}(u\rho, \mathsf{pub}(v\rho)) = (t_1\rho){\downarrow}$; and
- $(t_2{\downarrow})\rho = v\rho = (t_2\rho){\downarrow}$.

Therefore, we have that:

- $(t{\downarrow})\rho = (\mathsf{adec}(t_1, t_2){\downarrow})\rho = (\mathsf{adec}(t_1{\downarrow}, t_2{\downarrow}){\downarrow}\rho = u{\downarrow}\rho = u\rho$; and
- $(t\rho){\downarrow} = (\mathsf{adec}(t_1, t_2)\rho){\downarrow} = \mathsf{adec}(t_1\rho{\downarrow}, t_2\rho{\downarrow}){\downarrow} = u\rho$.

This allows us to conclude when $\mathsf{g} = \mathsf{adec}$, and the other cases can be done in a similar way.

Item 2. Let $t \in \mathcal{T}(\Sigma, \mathcal{C}_0^+ \uplus \mathcal{N})$ be a term in normal form and such that t is not a $\mathcal{C}_0^+$-message. In case $t \in \mathcal{T}(\Sigma_{\mathsf{c}}, \mathcal{C}_0^+ \uplus \mathcal{N})$, the only case where $t\rho$ is a message (whereas t is not) is if there is a constant c of sort $\mathsf{bitstring}$ such that $\mathsf{pub}(c)$ occurs in t, and $\rho(c)$ is of sort atom. Let c_0 be such a constant, and ρ be a $\mathcal{C}_0^+$-renaming such that $c_0 \notin dom(\rho) \cup img(\rho)$, we have that $t\rho{\downarrow}$ is not a $\mathcal{C}_0^+$-message since $\mathsf{pub}(c_0)$ occurs in $t\rho$ as a subterm and c_0 is not of sort atom.

Now, t must contain at least one symbol in $\Sigma_{\mathsf{d}} \cup \{\mathsf{check}\}$. Let p be one of the lowest positions such that $t = C[\mathsf{g}(t_1, \ldots, t_k)]_p$ for some $\mathsf{g} \in \Sigma_{\mathsf{d}} \cup \{\mathsf{check}\}$ and $t_1, \ldots, t_k \in \mathcal{T}(\Sigma_{\mathsf{c}}, \mathcal{C}_0^+ \uplus \mathcal{N})$. Let $u = \mathsf{g}(t_1, \ldots, t_k)$, and ρ_0 be a special renaming that maps any constant in $\mathcal{C}_0^+$ to $c_0 \in \mathcal{C}_0^+$ (i.e. an arbitrary constant of sort atom). Let $\ell \to r$ be the rewriting rule associated to g. Either the rule does not apply on $u\rho_0$, and thus the rule will not apply on $u\rho$ for any $\mathcal{C}_0^+$-renaming ρ. Otherwise, we have that the rule can be applied on $u\rho_0$ whereas it can not be applied on u. In such a case, we have that there are two positions $p_1 \neq p_2$ such that $\ell|_{p_1} = \ell|_{p_2} \in \mathcal{X}$, and thus $u|_{p_1} \neq u|_{p_2}$ whereas $(u\rho_0)|_{p_1} = (u\rho_0)|_{p_2}$. Moreover, we know that $u|_{p_1}$ and $u|_{p_2}$ are both constants in $\mathcal{C}_0^+$ (they are of sort atom since otherwise reduction is not possible). Let $c_1 = u|_{p_1}$. Any $\mathcal{C}_0^+$-renaming ρ with $c_1 \notin (dom(\rho) \cup img(\rho))$ will prevent the rewriting rule $\ell \to r$ to be applicable on $u\rho$ and thus on $t\rho$. This allows us to conclude that $t\rho{\downarrow}$ is not a $\mathcal{C}_0^+$-message.

Item 3. Let $t_1, t_2 \in \mathcal{T}(\Sigma, \mathcal{C}_0^+ \uplus \mathcal{N})$ such that $t_1{\downarrow}, t_2{\downarrow}$ are $\mathcal{C}_0^+$-messages and $t_1{\downarrow} \neq t_2{\downarrow}$. Thanks to Item 1, we have that $(t_1{\downarrow})\rho = (t_1\rho){\downarrow}$ and $(t_2{\downarrow})\rho = (t_2\rho){\downarrow}$. Therefore, we can simply show that if t_1, t_2 are $\mathcal{C}_0^+$-messages and $t_1 \neq t_2$ then there exists $c_0 \in \mathcal{C}_0^+$ such that $t_1\rho \neq t_2\rho$ for any $\mathcal{C}_0^+$-renaming ρ such that $c_0 \notin dom(\rho) \cup img(\rho)$.

Base case: The only non trivial base case is when both t_1 and t_2 are in $\mathcal{C}_0^+$. Assume w.l.o.g. that $t_2 = c_0$. Since $t_1 \neq t_2$, we have that $t_1\rho \neq t_2\rho$ for any $\mathcal{C}_0^+$-renaming ρ such that $c_0 \notin dom(\rho) \cup img(\rho)$. The other base cases where either t_1 or t_2 is in $\mathcal{C}_0^+ \uplus \mathcal{N}$ are trivial and we may actually choose any $\mathcal{C}_0^+$-renaming ρ.

Inductive case: Now, in case t_1 and t_2 are not atomic, we distinguish two cases. In case they do not have the same function symbol at their root, we can choose any $\mathcal{C}_0^+$-renaming ρ, and the disequality between $t_1\rho$ and $t_2\rho$ will be preserved. Now, assume that $t_1 = \mathsf{f}(u_1, \ldots, u_k)$ and $t_2 = \mathsf{f}(v_1, \ldots, v_k)$ with $\mathsf{f} \in \Sigma_{\mathsf{c}}$. We know that $u_i \neq v_i$ for some $i \in \{1, \ldots, k\}$. We can apply our induction hypothesis to conclude that there exists c_0 such $u_i\rho \neq v_i\rho$ for any $\mathcal{C}_0^+$-renaming ρ such that $c_0 \notin dom(\rho) \cup img(\rho)$. Therefore, considering any $\mathcal{C}_0^+$-renaming that satisfies such a condition will allow us to conclude. □

Lemma 11. *Let $\mathcal{K}_{\mathcal{P}}$ be an initial $\mathcal{C}_0^+$-configuration type-compliant w.r.t. $(\mathcal{T}_{\mathcal{P}}, \delta_{\mathcal{P}})$, ρ be a type-preserving A-renaming such that $\mathsf{A} \subseteq \mathcal{C}_0^+ \setminus \mathcal{C}_{\mathcal{P}}$, and $\mathcal{K}_{\mathcal{P}} \xrightarrow{\mathsf{tr}} (\mathcal{P}'; \phi'; \sigma'; i')$ be a quasi-well-typed execution. We have that $\mathcal{K}_{\mathcal{P}} \xrightarrow{\mathsf{tr}\rho} (\mathcal{P}'; \phi'\rho; \sigma'\rho; i')$ is a quasi-well-typed execution.*

Proof. Let $\mathcal{K}_\mathcal{P} = (\mathcal{P}; \emptyset; \emptyset; 0)$, and $K'_\mathcal{P} = (\mathcal{P}'; \phi'; \sigma'; i')$. We show this result by induction on the length n of the execution trace $\mathcal{K}_\mathcal{P} \xrightarrow{\mathsf{tr}} \mathcal{K}'_\mathcal{P}$.
Base case. We have that $\mathcal{K}'_\mathcal{P} = \mathcal{K}_\mathcal{P}$, and the result trivially holds.
Induction case. In such a case, we have that $\mathsf{tr} = \mathsf{tr}^- \cdot \alpha$, and we have that:

$$\mathcal{K}_\mathcal{P} \xrightarrow{\mathsf{tr}^-} (\mathcal{P}^-; \phi^-; \sigma^-; i^-) \xrightarrow{\alpha} (\mathcal{P}'; \phi'; \sigma'; i').$$

Relying on our induction hypothesis, we know that $\mathcal{K}_\mathcal{P} \xrightarrow{\mathsf{tr}^-} (\mathcal{P}^-\rho; \phi^-\rho; \sigma^-\rho; i^-)$. We distinguish three cases depending on α.

- Case $\alpha = \mathsf{phase}\ i'$. In such a case, we have that $\mathcal{P}' = \mathcal{P}^-$, $\sigma' = \sigma^-$, $\phi' = \phi^-$, and $i^- < i'$. We have that $(\mathcal{P}^-; \phi^-\rho; \sigma^-\rho; i^-) \xrightarrow{\mathsf{phase}\ i'} (\mathcal{P}^-; \phi^-\rho; \sigma^-\rho; i')$, and this allows us to conclude since $(\mathcal{P}^-; \phi^-\rho; \sigma^-\rho; i') = (\mathcal{P}'; \phi'\rho; \sigma'\rho; i')$.
- Case $\alpha = \mathsf{out}(c, \mathsf{w})$. In such a case, we have that $\mathcal{P}^- = \{i^-{:}\mathsf{out}(c, u).P_0\} \cup \mathcal{P}_0$ for some u, P_0, and $\mathcal{P}_0$. We have also that $\sigma' = \sigma^-$, $\phi' = \phi^- \cup \{\mathsf{w} \triangleright u\sigma^-\}$ and $i' = i^-$. To conclude that $(\mathcal{P}^-; \phi^-\rho; \sigma^-\rho; i^-) \xrightarrow{\mathsf{out}(c,\mathsf{w})} (\mathcal{P}'; \phi'\rho; \sigma'\rho; i')$, it is sufficient to show that $(u\sigma^-)\rho = u(\sigma^-\rho)$. Actually, $(u\sigma^-)\rho = (u\rho)(\sigma^-\rho)$, and since $dom(\rho) \subseteq \mathcal{C}_0^+ \setminus \mathcal{C}_\mathcal{P}$, we deduce that $u\rho = u$, and since $\rho(a)$ is an atom when a is an atom, this allows us to conclude.
- Case $\alpha = \mathsf{in}(c, R)$. In such a case, we have that $\mathcal{P}^- = \{i^-{:}\mathsf{in}(c, u).P_0\} \cup \mathcal{P}_0$ for some u, P_0, and $\mathcal{P}_0$. We have also that $\phi' = \phi^-$, $\sigma' = \sigma^- \uplus \sigma_0$ for some σ_0 such that $R\phi^-\!\downarrow = (u\sigma^-)\sigma_0$ and $R\phi^-\!\downarrow$ is a message. Moreover, $i^- = i'$. To conclude that $(\mathcal{P}^-; \phi^-\rho; \sigma^-\rho; i^-) \xrightarrow{\mathsf{in}(c,R\rho)} (\mathcal{P}'; \phi'\rho; \sigma'\rho; i')$, it remains to show that $(R\rho)(\phi^-\rho)\!\downarrow = u(\sigma^-\rho)\sigma'_0$ and $\sigma'\rho = \sigma^-\rho \uplus \sigma'_0$ for some σ'_0. Since $R\phi^-\!\downarrow = (u\sigma^-)\sigma_0$, we deduce that $(R\phi^-\!\downarrow)\rho = ((u\sigma^-)\sigma_0)\rho$, and thanks to Lemma 10 (item 1), we have that $(R\phi^-)\rho\!\downarrow = ((u\rho)(\sigma^-\rho))(\sigma_0\rho)$. Lastly, since ρ is an A-renaming and $\mathsf{A} \subseteq \mathcal{C}_0 \setminus \mathcal{C}_\mathcal{P}$, we know that $u\rho = u$, and therefore we have that $(R\rho)(\phi^-\rho)\!\downarrow = (u(\sigma^-\rho))(\sigma_0\rho)$. Moreover, since $\sigma' = \sigma^- \uplus \sigma_0$, we have that $\sigma'\rho = \sigma^-\rho \uplus \sigma_0\rho$. Therefore, choosing $\sigma'_0 = \sigma_0\rho$ allows us to conclude.

Since ρ is type-preserving, the resulting execution is quasi-well-typed when the original one is quasi-well-typed. □

Theorem 1. *Let $\mathcal{K}_\mathcal{P}$ be an initial $\mathcal{C}_0$-configuration type-compliant w.r.t. $(\mathcal{T}_\mathcal{P}, \delta_\mathcal{P})$ and $\mathcal{K}_\mathcal{Q}$ be another initial $\mathcal{C}_0$-configuration. Let $\mathcal{C}^\star = (\mathcal{C}_\mathcal{P} \cup \mathcal{C}_\mathcal{Q}) \uplus \{c_0^\star, c_1^\star, c_+^\star\}$. We have that $\mathcal{K}_\mathcal{P} \not\sqsubseteq_t \mathcal{K}_\mathcal{Q}$ w.r.t. $\mathcal{C}_0$ if, and only if, there exists a witness $(\mathsf{tr}, \phi) \in \mathsf{trace}(\mathcal{K}_\mathcal{P})$ of this non-inclusion which only involves constants from $\mathcal{C}^\star$, simple recipes, and with a quasi-well-typed underlying execution.*

Proof. The $\Leftarrow$ direction is actually easy to establish. It is a direct consequence of Lemma 3.11 established and proved in [17]. Therefore, we consider the other direction ($\Rightarrow$).

First, we apply Theorem 3 to obtain a witness $(\mathsf{tr}, \phi) \in \mathsf{trace}_{\mathcal{C}_0^+}(\mathcal{K}_\mathcal{P})$ of this non-inclusion which only involves simple recipes, and with a well-typed underlying execution. Then, considering $\mathcal{C}^{\star\star} = (\mathcal{C}_\mathcal{P} \cup \mathcal{C}_\mathcal{Q}) \uplus \{c \in \mathcal{C}_0^+ \mid \delta_\mathcal{P}(c) = \tau^\star\}$,

and applying an alpha-renaming along the derivation to rename constants from $\mathcal{C}_0^+ \setminus \mathcal{C}^{\star\star}$ to constants from $\mathcal{C}^{\star\star}$, we easily derive a witness which only involves simple recipes built using constants from $\mathcal{C}^{\star\star}$, and with a quasi-well-typed underlying execution.

We consider a minimal (in length) witness of non-inclusion, i.e. a trace $(\mathsf{tr}, \phi) \in \mathsf{trace}_{\mathcal{C}^{\star\star}}(\mathcal{K}_\mathcal{P})$ such that:

1. either $(\mathsf{tr}, \psi) \notin \mathsf{trace}_{\mathcal{C}^{\star\star}}(\mathcal{K}_\mathcal{Q})$ for any ψ; or
2. $(\mathsf{tr}, \psi) \in \mathsf{trace}_{\mathcal{C}^{\star\star}}(\mathcal{K}_\mathcal{Q})$ but $\phi \not\sqsubseteq_s \psi$.

We will apply a renaming on such a witness to get rid of constants from $\mathcal{C}^{\star\star} \setminus (\mathcal{C}_\mathcal{P} \cup \mathcal{C}_\mathcal{Q})$. Such a renaming ρ will be such that $dom(\rho) \subseteq \mathcal{C}^{\star\star} \setminus (\mathcal{C}_\mathcal{P} \cup \mathcal{C}_\mathcal{Q})$, and $img(\rho) \subseteq \{c_0^\star, c_1^\star, c_+^\star\}$. Moreover, in case c is of sort atom, then $\rho(c)$ will be of sort atom.

In the following, we will consider different renamings. In particular, we consider ρ_0 which maps any constant from $\mathcal{C}^{\star\star}$ to $c_0^\star$, as well as the renaming ρ_1 which maps any constant from $\mathcal{C}^{\star\star}$ of sort atom to $c_0^\star$, and any constant from $\mathcal{C}^{\star\star}$ of sort $\mathsf{bitstring}$ to $c_+^\star$.

Case 1: $\mathsf{tr} = \mathsf{tr}^- \cdot \alpha$ does not pass in $\mathcal{K}_\mathcal{Q}$. In such a case, we have that:

- $\mathcal{K}_\mathcal{P} \xrightarrow{\mathsf{tr}^-} (\mathcal{P}^-; \phi^-; \sigma_\mathcal{P}^-; i^-) \xrightarrow{\alpha} (\mathcal{P}; \phi; \sigma_\mathcal{P}; i)$;
- $\mathcal{K}_\mathcal{Q} \xrightarrow{\mathsf{tr}^-} (\mathcal{Q}^-; \psi^-; \sigma_\mathcal{Q}^-; i^-)$; and
- $\phi^- \sqsubseteq_s \psi^-$.

Relying on Lemma 11, we have that:

- $\mathcal{K}_\mathcal{P} \xrightarrow{\mathsf{tr}^-\rho} (\mathcal{P}^-; \phi^-\rho; \sigma_\mathcal{P}^-\rho; i^-) \xrightarrow{\alpha\rho} (\mathcal{P}; \phi\rho; \sigma_\mathcal{P}\rho; i)$; and
- $\mathcal{K}_\mathcal{Q} \xrightarrow{\mathsf{tr}^-\rho} (\mathcal{Q}^-; \psi^-\rho; \sigma_\mathcal{Q}^-\rho; i^-)$

for any A-renaming ρ such that $\mathsf{A} \subseteq \mathcal{C}_0^+ \setminus (\mathcal{C}_\mathcal{P} \cup \mathcal{C}_\mathcal{Q})$. Thus, to conclude, it remains to justify that $\alpha\rho$ can not be triggered from $(\mathcal{Q}^-; \psi^-\rho; \sigma_\mathcal{Q}^-; i^-)$ for a $\mathcal{C}_0^+$-renaming ρ such that $img(\rho) \subseteq \{c_0^\star, c_1^\star, c_+^\star\}$. We consider three cases depending on the action α.

1. *Case* $\alpha = \mathsf{phase}\ i$. In such a case, we have that $(\mathcal{Q}^-; \psi^-; \sigma_\mathcal{Q}^-; i^-) \not\xrightarrow{\mathsf{phase}\ i}$ and this is impossible since α can be triggered from $(\mathcal{P}^-; \phi^-; \sigma_\mathcal{P}^-; i^-)$, and thus $i^- < i$.
2. *Case* $\alpha = \mathsf{out}(c, \mathsf{w})$. In such a case, we have that $\mathcal{P}^- = \mathsf{out}(c, u).P_0 \uplus \mathcal{P}_0$. Either $\mathcal{Q}^-$ is not ready to perform an output on channel c, and thus this is the case for $(\mathcal{Q}^-; \psi^-\rho_0; \sigma_\mathcal{Q}^-\rho_0; i^-)$, and we are done. Otherwise, we have that $\mathcal{Q}^- = \mathsf{out}(c, v).Q_0 \uplus \mathcal{Q}_0$ but $v\sigma_\mathcal{Q}^-$ is not a $\mathcal{C}^{\star\star}$-message. We have that $(v\sigma_\mathcal{Q}^-)\rho_1 = v(\sigma_\mathcal{Q}^-\rho_1)$ is still not a message, and we are done.
3. *Case* $\alpha = \mathsf{in}(c, R)$. In such a case, we have that $\mathcal{P}^- = \mathsf{in}(c, u).P_0 \uplus \mathcal{P}_0$. Either $\mathcal{Q}^-$ is not ready to perform an imput on channel c, and thus $(\mathcal{Q}^-; \psi^-; \sigma_\mathcal{Q}^-; i^-)$ is not ready to perform an input on channel c, and we are done. Otherwise,

since $\phi^- \sqsubseteq_s \psi^-$, and $R\phi^-\downarrow$ is a message, we deduce that $R\psi^-\downarrow$ is a message, and we have that $R\psi^-\downarrow\rho_0 = (R\psi^-)\rho_0\downarrow = (R\rho_0)(\psi^-\rho_0)\downarrow$ is a message (Lemma 10 - item 1). Hence, since the input can not be triggered, this is due to a problem of filtering, there does not exist τ such that $(v\sigma^-_{\mathcal{Q}})\tau = R\psi^-\downarrow$. If $R\psi^-\downarrow$ and $v\sigma^-_{\mathcal{Q}}$ do not match because their structure differ, then no renaming will change that. Therefore, we consider the renaming ρ_0, and it is easy to see that $(R\rho_0)(\psi^-\rho_0)\downarrow$ and $v(\sigma^-\mathcal{Q}\rho_0)$ do not match. Otherwise, the only possibility is that there are two position p_1 and p_2 in $(v\sigma^-_{\mathcal{Q}})$ such that $(v\sigma^-_{\mathcal{Q}})|_{p_1} = (v\sigma^-_{\mathcal{Q}})|_{p_2} \in \mathcal{X}$, but $(R\psi^-\downarrow)|_{p_1} \neq (R\psi^-\downarrow)|_{p_2}$. Thanks to Lemma 10 (item 3), there is a constant c_0 such that for any $\mathcal{C}_0^+$-renaming ρ with $c_0 \notin dom(\rho) \cup img(\rho)$, $t_1\rho \neq t_2\rho$. In case $c_0 \in \mathcal{C}_{\mathcal{P}} \cup \mathcal{C}_{\mathcal{Q}}$, then we can simply consider ρ_0 that will kept c_0 unchanged. Otherwise, we assume w.l.o.g. that c_0 is $c_0^\star$ (or $c_+^\star$) (performing an alpha-renaming), and we consider the renaming ρ which maps any constants from $\mathcal{C}^{\star\star}$ (except $c_0^\star$ or $c_+^\star$) to c_1^+.

<u>*Case 2:*</u> We have that $\mathcal{K}_{\mathcal{P}} \xrightarrow{\mathsf{tr}} (\mathcal{P}; \phi; \sigma_{\mathcal{P}}; i)$; $\mathcal{K}_{\mathcal{Q}} \xrightarrow{\mathsf{tr}} (\mathcal{Q}; \psi; \sigma_{\mathcal{Q}}; i)$; and $\phi \not\sqsubseteq_s \psi$. Relying on Lemma 11, we have that:

- $\mathcal{K}_{\mathcal{P}} \xrightarrow{\mathsf{tr}\rho} (\mathcal{P}; \phi\rho; \sigma_{\mathcal{P}}\rho; i)$; and
- $\mathcal{K}_{\mathcal{Q}} \xrightarrow{\mathsf{tr}\rho} (\mathcal{Q}; \psi\rho; \sigma_{\mathcal{Q}}\rho; i)$

for any A-renaming ρ such that $\mathsf{A} \subseteq \mathcal{C}_0^+ \setminus (\mathcal{C}_{\mathcal{P}} \cup \mathcal{C}_{\mathcal{Q}})$. Thus, to conclude, it remains to justify that $\phi\rho \not\sqsubseteq \psi\rho$ for a $\mathcal{C}_0^+$-renaming ρ such that $img(\rho) \subseteq \{c_0^\star, c_1^\star, c_+^\star\}$. We distinguish two cases depending on the form of the test.

1. *Case message.* There is a recipe R w.r.t. $\mathcal{C}^{\star\star}$ such that $R\phi\downarrow$ is a $\mathcal{C}_0^+$-message whereas $R\psi\downarrow$ is not. Therefore, by Lemma 10 (item 2), there is a constant $c_0 \in \mathcal{C}_0^+$ such that for any $\mathcal{C}_0^+$-renaming ρ such that $c_0 \notin dom(\rho) \cup img(\rho)$, we have that $(R\psi)\rho\downarrow$ is not a message. In case $c_0 \in \mathcal{C}_{\mathcal{P}} \cup \mathcal{C}_{\mathcal{Q}}$, then we consider the renaming ρ_0. We have that $(R\rho_0)(\phi\rho_0)\downarrow = (R\phi)\rho_0\downarrow = R\phi\downarrow\rho_0$ (Lemma 10 - item 1) is a message. Moreover, we have seen that $(R\rho_0)(\psi\rho_0)\downarrow = (R\psi)\rho_0\downarrow$ is not a message. This allows us to conclude. Now, in case $c_0 \notin \mathcal{C}_{\mathcal{P}} \cup \mathcal{C}_{\mathcal{Q}}$, we assume w.l.o.g. that c_0 is $c_0^\star$ (or $c_+^\star$) (performing an alpha-renaming, and we consider the renaming ρ which maps any constants from $\mathcal{C}^{\star\star}$ (except $c_0^\star$ or $c_+^\star$) to c_1^+. We have that $(R\rho)(\phi\rho)\downarrow = (R\phi)\rho\downarrow = R\phi\downarrow\rho$ (Lemma 10 - item 1) is a message. Moreover, we have seen that $(R\rho)(\psi\rho)\downarrow = (R\psi)\rho\downarrow$ is not a message. Therefore, we have our witness of non-inclusion.
2. *Case test.* There are two recipes R_1 and R_2 w.r.t. $\mathcal{C}^{\star\star}$ such that $R_1\phi\downarrow$, $R_2\phi\downarrow$ are messages, and $R_1\phi\downarrow = R_2\phi\downarrow$. Moreover, we have that $R_1\psi\downarrow$ and $R_2\psi\downarrow$ are mesages, but $R_1\psi\downarrow \neq R_2\psi\downarrow$. Thanks to Lemma 10 (item 3), there is a constant $c_0 \in \mathcal{C}_0^+$ such that for any $\mathcal{C}_0^+$-renaming ρ such that $c_0 \notin dom(\rho) \cup img(\rho)$, we have that $(R_1\psi)\rho\downarrow \neq (R_2\psi)\rho\downarrow$. In case $c_0 \in \mathcal{C}_{\mathcal{P}} \cup \mathcal{C}_{\mathcal{Q}}$, then we consider the renaming ρ_0. We have that $(R_i\rho_0)(\phi\rho_0)\downarrow = (R_i\phi)\rho_0\downarrow = R_i\phi\downarrow\rho_0$ (Lemma 10 - item 1) is a message ($i \in \{1, 2\}$). Similarly, we have that $(R_i\rho_0)(\psi\rho_0)\downarrow = (R_i\psi)\rho_0\downarrow = R_i\psi\downarrow\rho_0$ with $i \in \{1, 2\}$. This allows us to

conclude that $R_1\rho_0 \stackrel{?}{=} R_2\rho_0$ is a test that holds in $\phi\rho_0$ but not in $\psi\rho_0$. Now, in case $c_0 \notin \mathcal{C}_\mathcal{P} \cup \mathcal{C}_\mathcal{Q}$, we assume w.l.o.g. that c_0 is $c_0^\star$ (or $c_+^\star$) (performing an alpha-renaming, and we consider the renaming ρ which maps any constants from $\mathcal{C}^{\star\star}$ (except $c_0^\star$ or $c_+^\star$) to c_1^+. We have that $(R_i\rho)(\phi\rho)\downarrow = (R_i\phi)\rho\downarrow = R_i\phi\downarrow\rho$ (Lemma 10 - item 1) is a message ($i \in \{1,2\}$). Similarly, we have that $(R_i\rho)(\psi\rho)\downarrow = (R_i\psi)\rho\downarrow = R_i\psi\downarrow\rho$ with $i \in \{1,2\}$. This allows us to conclude that $R_1\rho \stackrel{?}{=} R_2\rho$ is a test that holds in $\phi\rho$ but not in $\psi\rho$. Therefore, we have our witness of non-inclusion. □

D From trace inclusion to planning

The goal of this section is to establish Theorem 2. Before starting to prove this theorem, we formally define the notion of flattening (see Section D.1) and the notion of concretization (see Section D.2) briefly described in Section 4.2.

D.1 Flattening

We explain how to formally compute the set of flattened rules from a given abstract rule r. For this, we start by explaining how to decompose a fact $\mathsf{att}(u, v)$.

Definition 8. *Given a term $u \in \mathcal{T}_0(\Sigma_\mathsf{c}, \mathcal{C} \uplus \mathcal{N} \uplus \mathcal{X})$, we say that u is* decomposable *when:*

- *either $u \in \mathcal{X}$ and $\delta_\mathcal{P}(u)$ is not an initial type;*
- *or $u \notin \mathcal{C} \uplus \mathcal{N} \uplus \mathcal{X}$.*

Intuitively, a variable of non initial type is decomposable since it may be instantiated by a non atomic term which, in turns, may have been obtained by composition. Given $\mathsf{att}(u, v)$ with u decomposable, and let $\mathsf{f} \in \Sigma_\mathsf{c}$ be such that $\delta_\mathcal{P}(u) = \mathsf{f}(\tau_1, \ldots, \tau_k)$, we define $\mathsf{split}(\mathsf{att}(u, v))$ as follows:

$$\mathsf{split}(\mathsf{att}(u, v)) = (\mathsf{f}; \{\mathsf{att}(x_1, y_1), \ldots, \mathsf{att}(x_k, y_k)\}; \sigma_\mathcal{P}; \sigma_\mathcal{Q})$$

where

- $x_1, \ldots, x_k$ are fresh variables of type $\tau_1, \ldots, \tau_k$ and $\sigma_\mathcal{P} = mgu(u, \mathsf{f}(x_1, \ldots, x_k))$;
- $y_1, \ldots, y_k$ are fresh variables, $\sigma_\mathcal{Q} = mgu(v, \mathsf{f}(y_1, \ldots, y_k))$.

Note that $\sigma_\mathcal{P}$ exists and is necessarily a quasi-well-typed substitution. By convention, we assume that $mgu(u, u') = \bot$ when u and u' are not unifiable.

Let r be an abstract rule of the form $\mathsf{Pre} \to \mathsf{Add}; \mathsf{Del}$ with $f = \mathsf{att}(u, v) \in \mathsf{Pre}$ such that u is decomposable and $\mathsf{split}(f) = (\mathsf{f}, S, \sigma_\mathcal{P}, \sigma_\mathcal{Q})$. The decomposition of r w.r.t. f, denoted $\mathsf{decom}(r, f)$, is defined as follows:

1. $\big((\mathsf{Pre} \setminus f) \cup S \to \mathsf{bad}\big)\sigma_\mathcal{P}$ in case $\sigma_\mathcal{Q} = \bot$;
2. $\big((\mathsf{Pre} \setminus f) \cup S \to \mathsf{Add}; \mathsf{Del}\big)(\sigma_\mathcal{P} \uplus \sigma_\mathcal{Q})$ otherwise.

Then, decomposition is applied recursively on each rule.

$$\mathsf{Flat}(r) = \mathsf{Flat}(\{\mathsf{decom}(r, f) \mid f = \mathsf{att}(u, v) \in \mathsf{Pre}(r) \text{ with } u \text{ decomposable}\}) \cup \{r\}$$

Lemma 12. *Let $r = \mathsf{Pre}, \mathsf{att}(u, v) \to \mathsf{Add}; \mathsf{Del}$ be an abstract rule. Let σ be a grounding substitution for r such that $\delta_{\mathcal{P}}(x\sigma) \preceq \delta_{\mathcal{P}}(x)$ for any $x \in vars_{\mathsf{left}}(r)$. Let C be a constructor context such that $u\sigma = C[u_1, \ldots, u_n]$ and $v\sigma = C[v_1, \ldots, v_n]$. There exists $r' = \mathsf{Pre}', \mathsf{att}(u'_1, v'_1), \ldots, \mathsf{att}(u'_n, v'_n) \to \mathsf{Add}'; \mathsf{Del}'$ in $\mathsf{Flat}(r)$, and σ' a grounding substitution for r' such that:*

1. *$\delta_{\mathcal{P}}(x\sigma') \preceq \delta_{\mathcal{P}}(x)$ for any $x \in vars_{\mathsf{left}}(r')$;*
2. *$(\mathsf{Pre}', \mathsf{Add}', \mathsf{Del}')\sigma' = (\mathsf{Pre}, \mathsf{Add}, \mathsf{Del})\sigma$; and*
3. *$\mathsf{att}(u, v)\sigma = \mathsf{att}(C[u'_1, \ldots, u'_n], C[v'_1, \ldots, v'_n])\sigma'$.*

For the next lemmas, we need to be more specific on how the fact bad has appeared. Therefore, from now on, we consider three facts instead: bad-proto, bad-flat, and bad-concrete. Moreover, we assume that in protocol rules, bad is replaced by bad-proto, in flattening rules, bad is replaced by bad-flat, and in concretization rules bad is replaced by bad-concrete. When the precise origin of the failure does not matter, we simply write bad (meaning one of the three cases above).

Lemma 13. *Let r be an abstract protocol rule, and $r' \in \mathsf{Flat}(r)$ be such that*

$$r' = \mathsf{Pre}, \mathsf{att}(u_1, v_1), \ldots, \mathsf{att}(u_n, v_n) \to \mathsf{Add}; \mathsf{Del}$$

with $\mathsf{bad\text{-}flat} \notin \mathsf{Add}$. There exists a constructor context C and a substitution τ such that $r\tau = \mathsf{Pre}, \mathsf{att}(C[u_1, \ldots, u_n], C[v_1, \ldots, v_n]) \to \mathsf{Add}; \mathsf{Del}$.

For the next lemma, we need to define the relation $=^{\mathsf{left}}$ on facts as $f_1 =^{\mathsf{left}} f_2$ if and only if $f_1 = \mathsf{att}(u, v)$, $f_2 = \mathsf{att}(u', v')$ with $u = u'$, or $f_1 = \mathsf{state}^{c}_{P,Q}(\sigma_P, \sigma_Q)$, $f_2 = \mathsf{state}^{c'}_{P',Q'}(\sigma'_P, \sigma'_Q)$ with $c = c'$, $P = P'$ and $\sigma_P = \sigma'_P$. We extend this definition to Pre.

Lemma 14. *Let $r = \mathsf{Pre}, \mathsf{att}(u, v) \to \mathsf{Add}; \mathsf{Del}$ be an abstract protocol rule. Let σ be a grounding substitution for $vars_{\mathsf{left}}(r)$ such that $\delta_{\mathcal{P}}(x\sigma) \preceq \delta_{\mathcal{P}}(x)$ for any $x \in vars_{\mathsf{left}}(r)$. Let C be a constructor context such that $u\sigma = C[u_1, \ldots, u_n]$, whereas v and $C[z_1, \ldots, z_n]$ are not unifiable.*

There exists $r' = \mathsf{Pre}', \mathsf{att}(u'_1, v_1), \ldots, \mathsf{att}(u'_n, v_n) \to \mathsf{bad}$ in $\mathsf{Flat}(r)$, and σ' a grounding substitution for $vars_{\mathsf{left}}(r)$ such that $u'_i\sigma' = u_i$ for any $i \in \{1, \ldots, n\}$, and $\mathsf{Pre}'\sigma' =^{\mathsf{left}} \mathsf{Pre}\sigma$.

Lemma 15. *Let r be an abstract protocol rule, and $r' \in \mathsf{Flat}(r)$ be such that*

$$r' = \mathsf{Pre}, \mathsf{att}(u_1, v_1), \ldots, \mathsf{att}(u_n, v_n) \to \mathsf{bad\text{-}flat}$$

There exists a constructor context C, a substitution τ, a term v, and two sets Add_0 and Del_0 such that $r\tau = \mathsf{Pre}, \mathsf{att}(C[u_1, \ldots, u_n], v) \to \mathsf{Add}_0; \mathsf{Del}_0$ but v does not unify with $C[z_1, \ldots, z_n]$.

D.2 Concretization

Given an abstract rule r, we denote $vars_{\mathsf{left}}(r)$ the variables occurring on the left (first parameter) of a predicate occurring in r, i.e.

- $vars_{\mathsf{left}}(\mathsf{att}(u, v)) = vars(u)$; and
- $vars_{\mathsf{left}}(\mathsf{state}^c_{P,Q}(\sigma_P, \sigma_Q)) = vars(img(\sigma_P))$.

Given a substitution σ grounding for r, the application of σ on an abstract state is the concrete state obtained by simply composing the substitutions, i.e.

$$st^c_{P,Q}(\sigma_P, \sigma_Q)\sigma = st^c_{P,Q}(\sigma \circ \sigma_P, \sigma \circ \sigma_Q).$$

Given an abstract protocol rule r, its positive concretization simply consists in all its instantiations that are quasi-well-typed w.r.t. the left side of the rule.

$$\begin{aligned}\mathsf{Concrete}^+(r) = \{r\sigma \mid\ & \sigma \text{ substitution such that } r\sigma \text{ only involves messages} \\ & \text{with constants in } \mathcal{C}^\star \text{ and } \delta_{\mathcal{P}}(x\sigma) \preceq \delta_{\mathcal{P}}(x) \text{ for any } x \in vars_{\mathsf{left}}(r)\}\end{aligned}$$

Similarly to the static case, we need to make sure that we can detect when P and Q are *not* in trace inclusion. For this, we consider additional rules that express when a step that can be performed on the left hand side cannot be mimicked on the right hand side.

Given an abstract protocol rule $r = \mathsf{Pre} \rightarrow \mathsf{Add}; \mathsf{Del}$, $\mathsf{Concrete}^-(r)$ is the set of planning rules that contains: $f_1, \ldots, f_k \rightarrow \mathsf{bad}$ for any sequence of facts $f_1, \ldots, f_k$ such that $f_1, \ldots, f_k$ left-unify with Pre with substitution σ_L and $u \in \mathcal{T}_0(\Sigma_{\mathsf{c}}, \mathcal{N} \cup \mathcal{C}^\star)$ for any $\mathsf{att}(u, v) \in \mathsf{Add}\sigma_L$, and such that one of the following conditions holds:

- $f_1, \ldots, f_k$ does not right-unify with Pre;
- $f_1, \ldots, f_k$ right-unify with Pre with substitution σ_R but $v \notin \mathcal{T}_0(\Sigma_{\mathsf{c}}, \mathcal{N} \cup \mathcal{C}^\star)$ for some $\mathsf{att}(u, v) \in \mathsf{Add}\sigma_R$.

D.3 Bounded planning

In this section, we state one of our main lemma that allows on to relate execution traces and planning paths, and we rely on the following definitions:

Let $\mathcal{K}_{\mathcal{P}} = (\mathcal{P}; \sigma_{\mathcal{P}}; \phi; i_P)$ and $\mathcal{K}_{\mathcal{Q}} = (\mathcal{Q}; \sigma_{\mathcal{Q}}; \psi; i_Q)$ be two configurations with $dom(\phi) = dom(\psi)$ and such that $i_P = i_Q$. The set of facts associated to $\mathcal{K}_{\mathcal{P}}$ and $\mathcal{K}_{\mathcal{Q}}$ w.r.t. a given set of constants $\mathcal{C}$ is defined as follows:

$$\begin{aligned}\mathsf{Fact}_{\mathcal{C}}(\mathcal{K}_{\mathcal{P}}, \mathcal{K}_{\mathcal{Q}}) = \{\mathsf{Phase}(i)\} \cup \mathsf{Fact}_{\mathcal{C}}(\phi, \psi) \cup \\ \{\mathsf{state}^c_{P,Q}(\sigma_P, \sigma_Q) \mid P \in \mathcal{P},\, Q \in \mathcal{Q} \text{ are basic processes on channel } c, \\ \sigma_P = \sigma_{\mathcal{P}}|_{fv(P)} \text{ and } \sigma_Q = \sigma_{\mathcal{Q}}|_{fv(Q)}\}\end{aligned}$$

Note that this definition is in line with the one provided in the core of the paper for protocols assuming that $\mathcal{P}$ represents the configuration $(\mathcal{P}; \emptyset; \emptyset; 0)$.

Definition 9. *Given two sets of facts S and S' such that $S = \mathsf{Fact}_{\mathcal{C}}(\mathcal{K}_{\mathcal{P}}, \mathcal{K}_{\mathcal{Q}})$ with $\mathcal{K}_{\mathcal{P}} = (\mathcal{P}; \phi; \sigma_P; i)$ and $\mathcal{K}_{\mathcal{Q}} = (\mathcal{Q}; \psi; \sigma_Q; i)$ with $dom(\phi) = dom(\psi)$, we write $\mathsf{Fact}_{\mathcal{C}}(\mathcal{K}_{\mathcal{P}}, \mathcal{K}_{\mathcal{Q}}) \uparrow S'$ when:*

– $\mathsf{Fact}_{\mathcal{C}}(\mathcal{K}_{\mathcal{P}}, \mathcal{K}_{\mathcal{Q}})$ *and* S' *coincide on states fact, and also the phase fact;*
– *for any* $\mathsf{att}(u, v) \in \mathsf{Fact}_{\mathcal{C}}(\mathcal{K}_{\mathcal{P}}, \mathcal{K}_{\mathcal{Q}})$, $\mathsf{att}(u, v) \in S'$*; and*
– *for any* $\mathsf{att}(u, v) \in S'$*, there exists a destructor-only* $\mathcal{C}$*-recipe* R *such that* $R\phi{\downarrow} = u$*, and* $R\psi{\downarrow} = v$*.*

Below, $\mathsf{nb}_{\mathsf{in}}(\mathsf{tr})$ (resp. $\mathsf{nb}_{\mathsf{out}}(\mathsf{tr})$) is the number of input actions (resp. output actions) occurring in tr whereas $\mathsf{max}_{\mathsf{phase}}(\mathsf{tr})$ is the maximal integer occurring in a phase instruction in tr.

Lemma 16. *Let* $\mathcal{P}$ *be a protocol type-compliant w.r.t.* $(\mathcal{T}_{\mathcal{P}}, \delta_{\mathcal{P}})$*, and* $\mathcal{Q}$ *be another protocol. Let* $\Theta = \langle \mathcal{F}act_0, \mathsf{Fact}_{\mathcal{C}^\star}(\mathcal{P}, \mathcal{Q}), \mathcal{R} \rangle$ *where*

$$\mathcal{R} = \mathsf{Concrete}^{+}(\mathsf{R}_{\mathsf{Ana}} \cup \mathsf{Flat}(\mathsf{Rule}(\mathcal{P}, \mathcal{Q}))) \cup \mathcal{R}^{\mathsf{phase}}(\mathcal{P}).$$

Let $(\mathsf{tr}, \phi) \in \mathsf{trace}(\mathcal{P})$ *w.r.t.* $\mathcal{C}^\star$ *for some* ϕ *such that:*

– tr *only contains simple recipes;*
– (tr, ϕ) *is quasi-well-typed w.r.t.* $(\mathcal{T}_{\mathcal{P}}, \delta_{\mathcal{P}})$*;*
– $(\mathsf{tr}, \psi) \in \mathsf{trace}(\mathcal{Q})$ *w.r.t.* $\mathcal{C}^\star$ *for some* ψ*.*

Then, there exists a planning path from $\mathsf{Fact}_{\mathcal{C}^\star}(\mathcal{P}, \mathcal{Q})$ *to some* S *such that* $\mathsf{Fact}(\mathcal{K}'_{\mathcal{P}}, \mathcal{K}'_{\mathcal{Q}}) \uparrow S$ *of length at most*

$$\mathsf{nb}_{\mathsf{in}}(\mathsf{tr}) + \mathsf{nb}_{\mathsf{out}}(\mathsf{tr}) + \mathsf{max}_{\mathsf{phase}}(\mathsf{tr}) + (\mathsf{nb}_{\mathsf{in}}(\mathsf{tr}) + |\mathsf{Key}(\mathsf{tr})|) \times \mathsf{depth}(\phi)$$

where $\mathcal{K}'_{\mathcal{P}}$ *(resp.* $\mathcal{K}'_{\mathcal{Q}}$*) is the resulting configuration starting from* $\mathcal{P}$ *(resp.* $\mathcal{Q}$*) and executing* tr*, and* $\{\mathsf{att}(R\phi{\downarrow}, R\psi{\downarrow}) \mid R \in \mathsf{Key}(\mathsf{tr})\} \subseteq S$*.*

Conversely, let π *be a planning path from* $\mathsf{Fact}_{\mathcal{C}^\star}(\mathcal{P}, \mathcal{Q})$ *to* S *such that* $\mathsf{bad} \notin S$*. Then, there exist a trace* tr*, and frames* ϕ *and* ψ *such that:*

– tr *only contains simple recipes;*
– $(\mathsf{tr}, \phi) \in \mathsf{trace}(\mathcal{P})$ *w.r.t.* $\mathcal{C}^\star$ *and is quasi-well-typed w.r.t.* $(\mathcal{T}_{\mathcal{P}}, \delta_{\mathcal{P}})$*;*
– $(\mathsf{tr}, \psi) \in \mathsf{trace}(\mathcal{Q})$ *w.r.t.* $\mathcal{C}^\star$*; and*
– $\mathsf{Fact}_{\mathcal{C}^\star}(\mathcal{K}'_{\mathcal{P}}, \mathcal{K}'_{\mathcal{Q}}) \uparrow S$ *where* $\mathcal{K}'_{\mathcal{P}}$ *(resp.* $\mathcal{K}'_{\mathcal{Q}}$*) is the resulting configuration starting from* $\mathcal{P}$ *(resp.* $\mathcal{Q}$*) and executing* tr*.*

Proof. We show the two directions separately.
($\Rightarrow$) We do the proof by induction on the execution $\mathcal{K}_{\mathcal{P}} \xrightarrow{\mathsf{tr}} \mathcal{K}'_{\mathcal{P}}$.
Base case. The empty planning path can be used to establish the result.
Inductive case. We have that $\mathsf{tr} = \mathsf{tr}'.\alpha$. Therefore, we consider $\mathcal{K}''_{\mathcal{P}}$ and $\mathcal{K}''_{\mathcal{Q}}$ such that $\mathcal{K}_{\mathcal{P}} \xrightarrow{\mathsf{tr}'} \mathcal{K}''_{\mathcal{P}} \xrightarrow{\alpha} \mathcal{K}'_{\mathcal{P}}$, and $\mathcal{K}_{\mathcal{Q}} \xrightarrow{\mathsf{tr}'} \mathcal{K}''_{\mathcal{Q}} \xrightarrow{\alpha} \mathcal{K}'_{\mathcal{Q}}$. Let $\mathcal{K}''_{\mathcal{P}} = (\mathcal{P}''; \phi''; \sigma''_{\mathcal{P}}; i'')$, $\mathcal{K}'_{\mathcal{P}} = (\mathcal{P}'; \phi'; \sigma'_{\mathcal{P}}; i')$, $\mathcal{K}''_{\mathcal{Q}} = (\mathcal{Q}''; \psi''; \sigma''_{\mathcal{Q}}; i'')$, and $\mathcal{K}'_{\mathcal{Q}} = (\mathcal{Q}'; \psi'; \sigma'_{\mathcal{Q}}; i')$.

We apply our induction hypothesis on tr' and we obtain a planning path π_0 of length at most

$$\mathsf{nb}_{\mathsf{in}}(\mathsf{tr}') + \mathsf{nb}_{\mathsf{out}}(\mathsf{tr}') + \mathsf{max}_{\mathsf{phase}}(\mathsf{tr}') + (\mathsf{nb}_{\mathsf{in}}(\mathsf{tr}') + |\mathsf{Key}(\mathsf{tr}')|) \times \mathsf{depth}(\phi'')$$

from $\mathsf{Fact}_{\mathcal{C}^\star}(\mathcal{K}_\mathcal{P}, \mathcal{K}_\mathcal{Q})$ to some S''. Moreover, we have that $\mathsf{Fact}_{\mathcal{C}^\star}(\mathcal{K}''_\mathcal{P}, \mathcal{K}''_\mathcal{Q}) \uparrow S''$, and $\mathsf{att}(R\phi\downarrow, R\psi\downarrow) \in S''$ for any $R \in \mathsf{Key}(\mathsf{tr}')$. We perform a case analysis on the action α.

Case $\alpha = \mathsf{phase}\ i'$. We have that $i' > i''$, $\mathcal{P}' = \mathcal{P}''$, $\phi' = \phi''$ and $\sigma'_\mathcal{P} = \sigma''_\mathcal{P}$. Similarly, we have $\mathcal{Q}' = \mathcal{Q}''$, $\psi' = \psi''$ and $\sigma'_\mathcal{Q} = \sigma''_\mathcal{Q}$. As $\mathsf{Fact}_{\mathcal{C}^\star}(\mathcal{K}''_\mathcal{P}, \mathcal{K}''_\mathcal{Q}) \uparrow S''$, we have that $\mathsf{Phase}(i'') \in S''$. Let $r_i = \mathsf{Phase}(i) \to \mathsf{Phase}(i+1); \mathsf{Phase}(i)$ with $i \in \mathbb{N}$. We have that $\pi = \pi_0.r_{i''}.\ldots.r_{i'-1}$ is a planning path from $\mathsf{Fact}_{\mathcal{C}^\star}(\mathcal{K}_\mathcal{P}, \mathcal{K}_\mathcal{Q})$ to some S', with $\mathsf{Fact}_{\mathcal{C}^\star}(\mathcal{K}'_\mathcal{P}, \mathcal{K}'_\mathcal{Q}) \uparrow S'$ and $\mathsf{att}(R\phi\downarrow, R\psi\downarrow) \in S'$ for any $R \in \mathsf{Key}(\mathsf{tr})$. Since $\phi' = \phi''$, and $\mathsf{tr} = \mathsf{tr}'.\mathsf{phase}\ i'$, the length of π is at most

$$\mathsf{nb}_\mathsf{in}(\mathsf{tr}) + \mathsf{nb}_\mathsf{out}(\mathsf{tr}) + \mathsf{max}_\mathsf{phase}(\mathsf{tr}) + (\mathsf{nb}_\mathsf{in}(\mathsf{tr}) + |\mathsf{Key}(\mathsf{tr})|) \times \mathsf{depth}(\phi).$$

Case $\alpha = \mathsf{out}(c, \mathsf{w})$. We have that there exist P'_c, and Q'_c, as well as u and v such that

- $\mathcal{P}'' = \{i'':\mathsf{out}(c,u).P'_c\} \uplus \mathcal{P}_0$, and $\mathcal{P}' = \{i'':P'_c\} \uplus \mathcal{P}_0$;
- $\mathcal{Q}'' = \{i'':\mathsf{out}(c,v).Q'_c\} \uplus \mathcal{Q}_0$, and $\mathcal{Q}' = \{i'':Q'_c\} \uplus \mathcal{Q}_0$.

Moreover, we have that $\phi' = \phi'' \uplus \{\mathsf{w} \triangleright u\sigma''_\mathcal{P}\}$, and $\psi' = \psi'' \uplus \{\mathsf{w} \triangleright v\sigma''_\mathcal{Q}\}$; and $\sigma'_\mathcal{Q} = \sigma''_\mathcal{P}$, as well as $\sigma'_\mathcal{Q} = \sigma''_\mathcal{Q}$, and $i' = i''$. Moreover, we know that $u\sigma''_\mathcal{P}$ and $v\sigma''_\mathcal{Q}$ are messages. Let r be the abstract protocol rule corresponding to this step. We have that $r \in \mathsf{Rule}(\mathcal{P}, \mathcal{Q})$ and this rule is of the form:

$$\mathsf{Phase}(i''), \mathsf{St}(P, Q) \to \mathsf{att}(u, v), \mathsf{St}(P', Q'); \mathsf{St}(P, Q)$$

Now, we consider the concrete instance that corresponds to the execution mentioned above, i.e. the one obtained by applying $\sigma'_\mathcal{P} \uplus \sigma'_\mathcal{Q}$. This allows us to conclude in one step, and since $\mathsf{nb}_\mathsf{out}(\mathsf{tr}) = \mathsf{nb}_\mathsf{out}(\mathsf{tr}') + 1$, $\mathsf{nb}_\mathsf{in}(\mathsf{tr}) = \mathsf{nb}_\mathsf{in}(\mathsf{tr}')$, $\mathsf{Key}(\mathsf{tr}) = \mathsf{Key}(\mathsf{tr}')$, and $\mathsf{depth}(\phi') \geq \mathsf{depth}(\phi'')$, we get a plan of length at most

$$\mathsf{nb}_\mathsf{in}(\mathsf{tr}) + \mathsf{nb}_\mathsf{out}(\mathsf{tr}) + \mathsf{max}_\mathsf{phase}(\mathsf{tr}) + (\mathsf{nb}_\mathsf{in}(\mathsf{tr}) + |\mathsf{Key}(\mathsf{tr})|) \times \mathsf{depth}(\phi).$$

Case $\alpha = \mathsf{in}(c, R)$. We know that R is a simple recipe, i.e. $R = C[R_1, \ldots, R_k]$, and we have that there exist P'_c, and Q'_c, as well as u and v such that:

- $\mathcal{P}'' = \{i'':\mathsf{in}(c,u).P'_c\} \uplus \mathcal{P}_0$, and $\mathcal{P}' = \{i'':P'_c\} \uplus \mathcal{P}_0$;
- $\mathcal{Q}'' = \{i'':\mathsf{in}(c,v).Q'_c\} \uplus \mathcal{Q}_0$, and $\mathcal{Q}' = \{i'':Q'_c\} \uplus \mathcal{Q}_0$.

Moreover, we have that $\phi' = \phi''$, $\psi' = \psi''$; $\sigma'_\mathcal{P} = \sigma''_\mathcal{P} \uplus mgu(R\phi''\downarrow, u\sigma''_\mathcal{P})$, and $\sigma'_\mathcal{Q} = \sigma''_\mathcal{Q} \uplus mgu(R\psi''\downarrow, v\sigma''_\mathcal{Q})$, and $i' = i''$. Moreover, we know that $u\sigma'_\mathcal{P}$ and $v\sigma'_\mathcal{Q}$ are messages. Applying Lemma 8 with $\mathcal{R} = \mathsf{Key}(\mathsf{tr})$ and $\mathcal{R}_0 = \mathsf{Key}(\mathsf{tr}')$, we get a plan π_1 to S'' such that $\{\mathsf{att}(R'\phi\downarrow, R'\psi\downarrow) \mid R' \in \mathsf{Key}(\mathsf{tr}')\} \subseteq S''$, and the length of π_1 is at most $|\mathsf{Key}(\{R_1, \ldots, R_k\}) \setminus \mathsf{Key}(\mathsf{tr}')| \times \mathsf{depth}(\phi'')$.

Let r be the abstract protocol rule corresponding to this step. We have that $r \in \mathsf{Rule}(\mathcal{P}, \mathcal{Q})$ and this rule is of the form:

$$\mathsf{Phase}(i''), \mathsf{St}(P, Q), \mathsf{att}(u, v) \to \mathsf{St}(P', Q'); \mathsf{St}(P, Q).$$

We know that $R\phi'\downarrow = u\sigma'_{\mathcal{P}}$ and $R\psi'\downarrow = v\sigma'_{\mathcal{Q}}$ with $R = C[R_1, \ldots, R_k]$. Therefore, we know that $u\sigma'_{\mathcal{P}} = C[R_1\phi'\downarrow, \ldots, R_k\phi'\downarrow]$ and $v\sigma'_{\mathcal{Q}} = C[R_1\psi'\downarrow, \ldots, R_k\psi'\downarrow]$. Thanks to Lemma 12, there exists a rule $r' \in \mathsf{Flat}(r)$:

$$r' = \mathsf{Pre}', \mathsf{att}(u'_1, v'_1), \ldots, \mathsf{att}(u'_k, v'_k) \to \mathsf{Add}'; \mathsf{Del}'$$

and σ' a grounding substitution for r' such that:

1. $\delta_{\mathcal{P}}(x\sigma') \preceq \delta_{\mathcal{P}}(x)$ for any $x \in vars_{\mathsf{left}}(r')$;
2. $(\mathsf{Pre}', \mathsf{Add}', \mathsf{Del}')\sigma' = (\mathsf{Pre}, \mathsf{Add}, \mathsf{Del})(\sigma'_{\mathcal{P}} \uplus \sigma'_{\mathcal{Q}})$; and
3. $\mathsf{att}(u\sigma'_{\mathcal{P}}, v\sigma'_{\mathcal{Q}}) = \mathsf{att}(C[u'_1, \ldots, u'_k], C[v'_1, \ldots, v'_k])\sigma'$.

We have that $\mathsf{att}(R_i\phi'\downarrow, R_i\psi'\downarrow) = \mathsf{att}(u'_i\sigma', v'_i\sigma')$ for $1 \leq i \leq k$. Thanks to Lemma 7 there is a plan π_2 (that we can apply after π_1) of length at most $\mathsf{depth}(\phi')$ containing only attacker rules such that $\mathsf{att}(R_i\phi'\downarrow, R_i\psi'\downarrow)$ is in the final state after π_2. We consider $r'\sigma' \in \mathsf{Concrete}^+(r')$, and we obtain the expected result considering the plan $\pi_0.\pi_1.\pi_2.r''$ of length at most

$$\begin{aligned}
&\mathsf{nb}_{\mathsf{in}}(\mathsf{tr}') + \mathsf{nb}_{\mathsf{out}}(\mathsf{tr}') + \mathsf{max}_{\mathsf{phase}}(\mathsf{tr}') + (\mathsf{nb}_{\mathsf{in}}(\mathsf{tr}') + |\mathsf{Key}(\mathsf{tr}')|) \times \mathsf{depth}(\phi'') \\
&+|\mathsf{Key}(\{R_1, \ldots, R_k\}) \setminus \mathsf{Key}(\mathsf{tr}')| \times \mathsf{depth}(\phi'') + \mathsf{depth}(\phi') + 1 \\
\leq\ &\mathsf{nb}_{\mathsf{in}}(\mathsf{tr}) + \mathsf{nb}_{\mathsf{out}}(\mathsf{tr}) + \mathsf{max}_{\mathsf{phase}}(\mathsf{tr}) + (\mathsf{nb}_{\mathsf{in}}(\mathsf{tr}) + |\mathsf{Key}(\mathsf{tr})|) \times \mathsf{depth}(\phi')
\end{aligned}$$

($\Leftarrow$) We show this result by induction on the length of the planning path.
Base case. Obvious.

Inductive case. We have a planning path $r_1, \ldots, r_n$. Thanks to our induction hypothesis, we know that the result holds for $r_1, \ldots, r_{n-1}$ leading to the state S_{n-1}, and therefore the existence of a trace tr' such that:

- tr' only contains simple recipes;
- $(\mathsf{tr}', \phi'') \in \mathsf{trace}(\mathcal{P})$ w.r.t. $\mathcal{C}^\star$ and is quasi-well-typed w.r.t. $(\mathcal{T}_{\mathcal{P}}, \delta_{\mathcal{P}})$;
- $(\mathsf{tr}', \psi'') \in \mathsf{trace}_{\mathcal{C}}(\mathcal{Q})$ w.r.t. $\mathcal{C}^\star$ for some ψ''; and
- $\mathsf{Fact}_{\mathcal{C}^\star}(\mathcal{K}'''_{\mathcal{P}}, \mathcal{K}''_{\mathcal{Q}}) \uparrow S$ where $\mathcal{K}''_{\mathcal{P}}$ (resp. $\mathcal{K}''_{\mathcal{Q}}$) is the resulting configuration starting from $\mathcal{P}$ (resp. $\mathcal{Q}$) and executing tr'.

We distinguish several cases depending on the rule r_n. In case r_n is an instance of $\mathsf{Concrete}^+(\mathsf{R}_{\mathsf{Ana}})$, we consider $\mathsf{tr} = \mathsf{tr}'$ again. The case where r_n is a rule that adds bad is impossible since $\mathsf{bad} \notin S_n$. The case where r_n is $\mathsf{Phase}(i) \to \mathsf{Phase}(i+1); \mathsf{Phase}(i)$ can be mimicked in the execution by the phase instruction. Now, if r_n is an instance of an abstract rule in $\mathsf{Flat}(\mathsf{Rule}(\mathcal{P}, \mathcal{Q}))$. Let r_f be the flattened abstract rule, and r the abstract protocol rule.

In case r is a rule corresponding to the case of an output on channel c, then r_n is an instance of r since the flattening does not produce any other rule. In such a case, we can mimick this step by considering $\mathsf{tr}'.\mathsf{out}(c, \mathsf{w})$.

In case r is a rule corresponding to an input, then r_n is an instance of a rule $r_f \in \mathsf{Flat}(r)$. We have that r_f is of the form:

$$\mathsf{Phase}(i), \mathsf{St}^{\mathsf{c}}_{P,Q}(\theta_P, \theta_Q), \mathsf{att}(u_1, v_1), \ldots, \mathsf{att}(u_k, v_k) \to \mathsf{St}^{\mathsf{c}}_{P',Q'}(\theta_{P'}, \theta_{Q'})$$

and r_n is an instance of $r_f(\sigma''_{\mathcal{P}} \cup \sigma''_{\mathcal{Q}})$ where $\sigma''_{\mathcal{P}}$ (resp. $\sigma''_{\mathcal{Q}}$) is the substitution obtained after executing tr'. Let τ_P, and τ_Q be grounding substitution such that $r_n = (r_f(\sigma''_{\mathcal{P}} \cup \sigma''_{\mathcal{Q}}))(\tau_P \cup \tau_Q)$.

We know that there exist destructor only recipes $R_1, \ldots, R_k$ such that $R_i\phi''{\downarrow} = u_i\sigma''_{\mathcal{P}}\tau_P$ and $R_i\psi''{\downarrow} = v_i\sigma''_{\mathcal{Q}}\tau_Q$ by Lemma 5. We can apply Lemma 13 on rule r_n written as:

$$r_n = \mathsf{Pre}, \mathsf{att}(u_1, v_1), \ldots, \mathsf{att}(u_n, v_n) \to \mathsf{Add}; \mathsf{Del}$$

We are in the case where $\mathsf{bad\text{-}flat} \notin \mathsf{Add}$, so there exists a constructor context C and a substitution τ such that $r_f(\sigma''_{\mathcal{P}} \cup \sigma''_{\mathcal{Q}})\tau = \mathsf{Pre}, \mathsf{att}(u, v) \to \mathsf{Add}; \mathsf{Del}$ where $u = C[u_1, \ldots, u_n]$ and $v = C[v_1, \ldots, v_n]$. Therefore, consider the trace $\mathsf{tr}' \cdot \mathsf{in}(c, C[R_1, \ldots, R_k])$. This step can be done on the $\mathcal{P}$ side, as well as on the $\mathcal{Q}$ side. Hence, the result. □

Theorem 2. *Let $\mathcal{P}$ a protocol type-compliant w.r.t. $(\mathcal{T}_{\mathcal{P}}, \delta_{\mathcal{P}})$, and $\mathcal{Q}$ be another protocol. We consider the following set $\mathcal{R}$ of concrete rules:*

$$\mathsf{Concrete}(\mathsf{R}_{\mathsf{Ana}} \cup \mathsf{Flat}(\mathsf{Rule}(\mathcal{P}, \mathcal{Q}))) \cup \mathcal{R}^{\mathsf{phase}} \cup \mathcal{R}^{\mathsf{test}_1}_{\mathsf{fail}} \cup \mathcal{R}^{\mathsf{test}_2}_{\mathsf{fail}} \cup \mathcal{R}^{\mathsf{atom}}_{\mathsf{fail}} \cup \mathcal{R}^{\mathsf{check}}_{\mathsf{fail}} \cup \mathcal{R}^{\mathsf{pub}}_{\mathsf{fail}}$$

Let $\Theta = \langle \mathcal{F}act_0, \mathsf{Fact}_{\mathcal{C}^\star}(\mathcal{P}, \mathcal{Q}), \mathcal{R}\rangle$ and $\Pi = \langle \Theta, \{\mathsf{bad}\}\rangle$. We have that $\mathcal{P} \not\sqsubseteq_t \mathcal{Q}$ if, and only if, Π has a solution of length

$$1 + \mathsf{nb}_{\mathsf{in}}(\mathcal{P}) + \mathsf{nb}_{\mathsf{out}}(\mathcal{P}) + \mathsf{max}_{\mathsf{phase}}(\mathcal{P}) + \mathsf{depth}(\delta_{\mathcal{P}}(\mathcal{P})) \times [1 + \mathsf{nb}_{\mathsf{in}}(\mathcal{P}) + N]$$

where N is the number of names occurring in $\mathcal{P}$ having a key type, i.e. such that $\delta_{\mathcal{P}}(n)$ (resp. $\mathsf{pub}(\delta_{\mathcal{P}}(n))$) occurs in key position of an encryption in $\delta_{\mathcal{P}}(\mathcal{P})$.

Proof. We show the two directions separately.

($\Rightarrow$) In case $\mathcal{P} \not\sqsubseteq_t \mathcal{Q}$, we know thanks to Theorem 1 that there exists a witness of this fact such that $(\mathsf{tr}, \phi) \in \mathsf{trace}(\mathcal{P})$ is quasi-well-typed, only involve the constants we consider here, and tr is made of simple recipes. We consider such a witness of minimal length, and then we consider one such that $\mathsf{Key}(\mathsf{tr})$ is minimal. By minimality of tr, we can assume that there is no $\mathsf{phase}\ i$ action in tr for $i > \mathsf{max}_{\mathsf{phase}}(\mathcal{P})$. We distinguish two cases depending on the fact that $(\mathsf{tr}, \psi) \in \mathsf{trace}(\mathcal{Q})$ for some ψ or not.

Case $(\mathsf{tr}, \psi) \in \mathsf{trace}(\mathcal{Q})$ for some ψ. Lemma 16 allows us to conclude that there exists a planning path π_0 from $\mathsf{Fact}_{\mathcal{C}^\star}(\mathcal{P}, \mathcal{Q})$ to S of length at most

$$\mathsf{nb}_{\mathsf{in}}(\mathsf{tr}) + \mathsf{nb}_{\mathsf{out}}(\mathsf{tr}) + \mathsf{max}_{\mathsf{phase}}(\mathsf{tr}) + (\mathsf{nb}_{\mathsf{in}}(\mathsf{tr}) + |\mathsf{Key}(\mathsf{tr})|) \times \mathsf{depth}(\phi)$$

such that:

- $\mathsf{Fact}_{\mathcal{C}^\star}(\mathcal{K}'_{\mathcal{P}}, \mathcal{K}'_{\mathcal{Q}}) \uparrow S$ where $\mathcal{K}'_{\mathcal{P}}$ (resp. $\mathcal{K}'_{\mathcal{Q}}$) is the resulting configuration starting from $\mathcal{P}$ (resp. $\mathcal{Q}$) and executing tr; and
- $\{\mathsf{att}(R\phi{\downarrow}, R\psi{\downarrow}) \mid R \in \mathsf{Key}(\mathsf{tr})\} \subseteq S$.

In case $(\mathsf{tr}, \psi) \in \mathsf{trace}(\mathcal{Q})$, we know that $\phi \not\sqsubseteq_s^{\mathsf{simple}^+} \psi$. We consider a static inclusion test T w.r.t. $\sqsubseteq_s^{\mathsf{simple}^+}$ which is minimal for the following measure (ordered lexicographically):

– $\mathsf{Key}(T) \setminus \mathsf{Key}(\mathsf{tr})$;
– its size, i.e. number of function symbols occurring in T.

Lemma 9 applies with $\mathcal{R}_0 = \mathsf{Key}(\mathsf{tr})$. We deduce that there is a plan π_1 of bad of length at most

$$(|\mathsf{Key}(T) \setminus \mathsf{Key}(\mathsf{tr})| + 1) \times \mathsf{depth}(\phi) + 1$$

It gives us a plan $\pi_0.\pi_1$ of bad of length at most

$$\mathsf{nb}_{\mathsf{in}}(\mathsf{tr}) + \mathsf{nb}_{\mathsf{out}}(\mathsf{tr}) + \mathsf{max}_{\mathsf{phase}}(\mathsf{tr}) + (\mathsf{nb}_{\mathsf{in}}(\mathsf{tr}) + |\mathsf{Key}(\mathsf{tr}) \cup \mathsf{Key}(T)| + 1) \times \mathsf{depth}(\phi) + 1$$

Since $\mathsf{nb}_{\mathsf{in}}(\mathsf{tr}) \leq \mathsf{nb}_{\mathsf{in}}(\mathcal{P})$, $n_{\mathsf{out}}(\mathsf{tr}) \leq \mathsf{nb}_{\mathsf{out}}(\mathcal{P})$, $\mathsf{max}_{\mathsf{phase}}(\mathsf{tr}) \leq \mathsf{max}_{\mathsf{phase}}(\mathcal{P})$ (by minimality of tr), $\mathsf{depth}(\phi) \leq \mathsf{depth}(\delta_{\mathcal{P}}(\mathcal{P}))$, to conclude it remains to prove that $|\mathsf{Key}(\mathsf{tr}) \cup \mathsf{Key}(T)| \leq N$. First, we have that $\{R\phi{\downarrow} \mid R \in \mathsf{Key}(\mathsf{tr}) \cup \mathsf{Key}(T)\} \setminus \mathcal{C}_0 \leq N$ since all these recipes occur in key position (i.e. 2nd argument of $\mathsf{sdec}/\mathsf{adec}$) of a recipe that leads to a message in ϕ. Now, in case there are two recipes in $K, K' \in \mathsf{Key}(\mathsf{tr}) \cup \mathsf{Key}(T) \cup \mathcal{C}_0$ such that $K\phi{\downarrow} = K'\phi{\downarrow}$, then by minimality of our witness of non-inclusion, we know that $K\psi{\downarrow} = K'\psi{\downarrow}$, and therefore replacing K or K' with the one which is deducible earlier or the smallest one in case they are both deducible at the same stage will give us a smaller test according to our measure. So we get that the length is smaller than

$$\mathsf{nb}_{\mathsf{in}}(\mathcal{P}) + \mathsf{nb}_{\mathsf{out}}(\mathcal{P}) + \mathsf{max}_{\mathsf{phase}}(\mathcal{P}) + [\mathsf{nb}_{\mathsf{in}}(\mathcal{P}) + N + 1] \times \mathsf{depth}(\delta_{\mathcal{P}}(\mathcal{P})) + 1$$

Case tr *does not pass in* $\mathcal{Q}$. Let $\mathsf{tr} = \mathsf{tr}^{-1}.\alpha$. We have that $(\mathsf{tr}^{-1}, \psi^{-1}) \in \mathsf{trace}(\mathcal{Q})$ but the last action α can not be performed. Lemma 16 allows us to conclude that there exists a planning path π_0 from $\mathsf{Fact}_{\mathcal{C}^\star}(\mathcal{P}, \mathcal{Q})$ to S of length at most

$$\mathsf{nb}_{\mathsf{in}}(\mathsf{tr}^{-1}) + \mathsf{nb}_{\mathsf{out}}(\mathsf{tr}^{-1}) + \mathsf{max}_{\mathsf{phase}}(\mathsf{tr}^{-1}) + (\mathsf{nb}_{\mathsf{in}}(\mathsf{tr}^{-1}) + |\mathsf{Key}(\mathsf{tr}^{-1})|) \times \mathsf{depth}(\phi)$$

such that:

– $\mathsf{Fact}_{\mathcal{C}^\star}(\mathcal{K}'_{\mathcal{P}}, \mathcal{K}'_{\mathcal{Q}}) \uparrow S$ where $\mathcal{K}'_{\mathcal{P}} = (\mathcal{P}'; \phi'; \sigma'_{\mathcal{P}}; i')$ (resp. $\mathcal{K}'_{\mathcal{Q}} = (\mathcal{Q}'; \psi'; \sigma'_{\mathcal{Q}}; i')$) is the resulting configuration starting from $\mathcal{P}$ (resp. $\mathcal{Q}$) and executing tr^{-1}; and
– $\{\mathsf{att}(R\phi{\downarrow}, R\psi{\downarrow}) \mid R \in \mathsf{Key}(\mathsf{tr}^{-1})\} \subseteq S$.

We consider three cases depending on the action α.

Case: $\alpha = \mathsf{phase}\ i$. This case is impossible since this action can be performed by $\mathcal{Q}$ as soon as its current phase is smaller than i, and this is the case since both $\mathcal{P}$ and $\mathcal{Q}$ are synchronised and the action α is feasible by $\mathcal{P}$.

Case: $\alpha = \mathsf{out}(c, \mathsf{w})$. If such an action can not be performed, it means that this action is not available in the process or would lead to output a term that is not a message. In the first case, we have an abstract protocol rule r that can be instantiated to mimick this step. In the second case, we have to consider the instance in $\mathsf{Concrete}^-(r)$. Note that for such a rule $\mathsf{Flat}(r) = r$. In each case, we only use one rule, so the plan is of length at most

$$\mathsf{nb}_{\mathsf{in}}(\mathsf{tr}^{-1}) + \mathsf{nb}_{\mathsf{out}}(\mathsf{tr}^{-1}) + \mathsf{max}_{\mathsf{phase}}(\mathsf{tr}^{-1}) + (\mathsf{nb}_{\mathsf{in}}(\mathsf{tr}^{-1}) + |\mathsf{Key}(\mathsf{tr}^{-1})|) \times \mathsf{depth}(\phi) + 1$$

As before, it is smaller than

$$\mathsf{nb_{in}}(\mathcal{P}) + \mathsf{nb_{out}}(\mathcal{P}) + \mathsf{max_{phase}}(\mathcal{P}) + \mathsf{depth}(\delta_{\mathcal{P}}(\mathcal{P})) \times [1 + \mathsf{nb_{in}}(\mathcal{P}) + N] + 1$$

Case: $\alpha = \mathsf{in}(c, R)$ with $R = C[R_1, \ldots, R_k]$ a simple recipe. We consider $\mathsf{in}(c, u)$ the corresponding action in $\mathcal{K}'_p$ and we have that $R\phi'{\downarrow} = (u\sigma'_{\mathcal{P}})\tau_{\mathcal{P}}$ for some substitution $\tau_{\mathcal{P}}$. If such an action can not be performed, it means that either this action is not syntactically available in the process or the term in the Q side does not match. Let r be the abstract protocol rule corresponding to this step.

Thanks to Lemma 8 applied with $\mathcal{R} = \mathsf{Key}(\{R_1, \ldots, R_k\})$ and $\mathcal{R}_0 = \mathsf{Key}(\mathsf{tr}^{-1})$, there is a plan π_1 of length at most $|\mathsf{Key}(\{R_1, \ldots, R_k\}) \setminus \mathsf{Key}(\mathsf{tr}^{-1})| \times \mathsf{depth}(\phi)$ of $\{\mathsf{att}(R''\phi{\downarrow}, R''\psi{\downarrow}) \mid R'' \in \mathsf{Key}(\{R_1, \ldots, R_k\})\}$. It allows us to apply Lemma 7 on $\mathcal{R} = \{R_1, \ldots, R_k\}$: there is a plan π_2 of length at most $\mathsf{depth}(\phi)$ of

$$\{\mathsf{att}(R_1\phi{\downarrow}, R_1\psi{\downarrow}), \ldots, \mathsf{att}(R_k\phi{\downarrow}, R_k\psi{\downarrow})\}.$$

Thus, with a similar reasoning as before, $\pi_0.\pi_1.\pi_2$ is a plan of length at most

$$\mathsf{nb_{in}}(\mathcal{P}) + \mathsf{nb_{out}}(\mathcal{P}) + \mathsf{max_{phase}}(\mathcal{P}) + \mathsf{depth}(\delta_{\mathcal{P}}(\mathcal{P})) \times [1 + \mathsf{nb_{in}}(\mathcal{P}) + N]$$

If the input is not syntactically available in the process, then r is of the form $\mathsf{Phase}(i), St(P, Q), \mathsf{att}(u, y) \to \mathsf{bad\text{-}proto}$ where y is a fresh variable and i an integer. $St(P, Q)$ unifies with the corresponding fact of S with substitution $\sigma'_{\mathcal{P}} \uplus \sigma'_{\mathcal{Q}} \uplus \tau_{\mathcal{P}}$ such that

- $St(P, Q)(\sigma'_{\mathcal{P}} \uplus \sigma'_{\mathcal{Q}} \uplus \tau_{\mathcal{P}}) \in S$.
- $R\phi'{\downarrow} = u(\sigma'_{\mathcal{P}} \uplus \tau_{\mathcal{P}})$
- $\delta_{\mathcal{P}}(x(\sigma'_{\mathcal{P}} \uplus \tau_{\mathcal{P}})) \preceq \delta_{\mathcal{P}}(x)$ for any $x \in vars_{\mathsf{left}}(r)$.

We can apply Lemma 12 on r and $\sigma'_{\mathcal{P}} \uplus \sigma'_{\mathcal{Q}} \uplus \tau_{\mathcal{P}} \uplus \{y \mapsto R\psi'{\downarrow}\}$ (Note that $R\psi'{\downarrow}$ is a message by minimality of our witness tr). We get a rule $r_f \in \mathsf{Flat}(r)$:

$$r_f = \mathsf{Pre}', \mathsf{att}(u'_1, v'_1), \ldots, \mathsf{att}(u'_k, v'_k) \to \mathsf{Add}'; \mathsf{Del}'$$

and a grounding substitution σ' such that:

- $\delta_{\mathcal{P}}(x\sigma') \preceq \delta_{\mathcal{P}}(x)$ for any $x \in vars_{\mathsf{left}}(r')$
- $(\mathsf{Pre}', \mathsf{Add}', \mathsf{Del}')\sigma' = (St(P, Q), \mathsf{bad\text{-}proto}, \emptyset)(\sigma'_{\mathcal{P}} \uplus \sigma'_{\mathcal{Q}} \uplus \tau_{\mathcal{P}} \uplus \{y \mapsto R\psi'{\downarrow}\})$
- $u(\sigma'_{\mathcal{P}} \uplus \tau_{\mathcal{P}}) = C[u'_1, \ldots, u'_k]$ and $R\psi'{\downarrow} = C[v'_1, \ldots, v'_k]$.

As the $\mathsf{att}(R_i\phi'{\downarrow}, R_i\psi'{\downarrow})$ unify with $\mathsf{Pre}(r_f)$, there is a rule r' in $\mathsf{Concrete}^+(r_f)$ such that its preconditions are exactly the $\mathsf{att}(R_i\phi'{\downarrow}, R_i\psi'{\downarrow})$. This concludes the case where the input is not syntactically available in the process.

The cases where there is an input $\mathsf{in}(c, v)$ in the Q side can be done in a rather similar way. We rely on Lemma 12 when v unifies with $C[z_1, \ldots, z_k]$, and we have a plan reaching bad applying a concrete rule in $\mathsf{Concrete}^-(\mathsf{Flat}(r))$. We rely on Lemma 14 otherwise, and the concrete planning rule leading to bad is either in $\mathsf{Concrete}^+(\mathsf{Flat}(r))$ or in $\mathsf{Concrete}^-(\mathsf{Flat}(r))$.

In each case, we conclude with a plan of bad of length at most

$$\mathsf{nb}_{\mathsf{in}}(\mathcal{P}) + \mathsf{nb}_{\mathsf{out}}(\mathcal{P}) + \mathsf{max}_{\mathsf{phase}}(\mathcal{P}) + \mathsf{depth}(\delta_{\mathcal{P}}(\mathcal{P})) \times [1 + \mathsf{nb}_{\mathsf{in}}(\mathcal{P}) + N] + 1$$

($\Leftarrow$) We show this result by induction on the length of planning path leading to bad. We consider one without rules in parallel of minimal length. We have that $\pi = r_1, \ldots, r_n$, and we denote S_i the state obtained after executing $r_1, \ldots, r_i$. We apply Lemma 16 on $r_1, \ldots, r_{n-1}$. We obtain the trace tr', and frames ϕ' nad ψ' such that:

- tr' only contains simple recipes;
- $(\mathsf{tr}', \phi') \in \mathsf{trace}(\mathcal{P})$ w.r.t. $\mathcal{C}^\star$ and is quasi-well-typed w.r.t. $(\mathcal{T}_{\mathcal{P}}, \delta_{\mathcal{P}})$;
- $(\mathsf{tr}', \psi') \in \mathsf{trace}(\mathcal{Q})$ w.r.t. $\mathcal{C}^\star$; and
- $\mathsf{Fact}_{\mathcal{C}^\star}(\mathcal{K}'_{\mathcal{P}}, \mathcal{K}'_{\mathcal{Q}}) \uparrow S_{n-1}$ where $\mathcal{K}'_{\mathcal{P}} = (\mathcal{P}'; \phi'; \sigma'_{\mathcal{P}}; i')$ (resp. $\mathcal{K}'_{\mathcal{Q}} = (\mathcal{Q}'; \psi'; \sigma'_{\mathcal{Q}}; i')$) is the resulting configuration starting from $\mathcal{P}$ (resp. $\mathcal{Q}$) and executing tr.

Note that it implies that $\mathsf{Phase}(i) \in S_{n-1}$. Now, we distinguish several cases depending on whether r_n is

1. in $\mathsf{Concrete}^-(\mathsf{R}_{\mathsf{Ana}}) \cup \mathcal{R}^{\mathsf{test}_1}_{\mathsf{fail}} \cup \mathcal{R}^{\mathsf{test}_2}_{\mathsf{fail}} \cup \mathcal{R}^{\mathsf{atom}}_{\mathsf{fail}} \cup \mathcal{R}^{\mathsf{check}}_{\mathsf{fail}} \cup \mathcal{R}^{\mathsf{pub}}_{\mathsf{fail}}$; or
2. in $\mathsf{Concrete}(\mathsf{Flat}(\mathsf{Rule}(\mathcal{P}, \mathcal{Q}))$

Note that r_n cannot be a phase rule as it would not allows us to reach bad. In the first case, we conclude relying on Proposition 1, and therefore we obtain that the frames ϕ' and ψ' resulting from the execution of tr' are not in static inclusion. The second case occurs when $r_n \in \mathsf{Concrete}(\mathsf{Flat}(\mathsf{Rule}(\mathcal{P}, \mathcal{Q})))$. So there is an abstract protocol rule r_f such that $r_n \in \mathsf{Concrete}(\mathsf{Flat}(r_f))$. Similarly to the proof done in [20], we show that this "planning" step can be mimicked on the protocol side form the P side, and has no counterpart on the Q side. To prove this, we rely on Lemma 13 and Lemma 15 to establish that any planning rule obtained by flattening is linked to an abstract protocol rule, which in turn can be mimicked on the protocol side. □

E Examples of non termination

We exhibit here two examples on which the original SAT-Equiv algorithm does not terminate. Given a channel c, consider the processes $P(c)$ and $Q(c)$ defined as follows.

$$P(c) := \mathsf{in}(c, \langle x, \mathsf{a}\rangle).\mathsf{out}(c, \langle x, \mathsf{a}\rangle)$$
$$Q(c) := \mathsf{in}(c, \langle x, \mathsf{a}\rangle).\mathsf{out}(c, \langle\langle x, x\rangle, \mathsf{a}\rangle)$$

where a is a public constant and x a variable. We consider $\mathcal{K}_P = \{P(c_1); P(c_2)\}$ and $\mathcal{K}_Q = \{Q(c_1); Q(c_2)\}$ for some public channel names c_1, c_2. On this example, starting with $\mathsf{att}(\mathsf{b}, \mathsf{b})$ (b being simply a public constant in the initial knowledge of the attacker), the following facts will be successively added when computing the planning graph:

- $\mathsf{att}(\langle \mathsf{b}, \mathsf{a}\rangle, \langle \mathsf{b}, \mathsf{a}\rangle)$,

– $\mathsf{att}(\langle \mathsf{b}, \mathsf{a}\rangle, \langle\langle \mathsf{b}, \mathsf{b}\rangle, \mathsf{a}\rangle)$,

– $\mathsf{att}(\langle \mathsf{b}, \mathsf{a}\rangle, \langle\langle\langle \mathsf{b}, \mathsf{b}\rangle, \langle \mathsf{b}, \mathsf{b}\rangle\rangle, \mathsf{a}\rangle)$, ...

Actually, $\mathsf{att}(\langle \mathsf{b}, \mathsf{a}\rangle, \langle\langle \mathsf{b}, \mathsf{b}\rangle, \mathsf{a}\rangle)$ can be added in two different ways: either considering the output on c_1, or the one on c_2. Therefore this fact will not be put in mutex with the other ones. In particular, the fact $\mathsf{att}(\langle \mathsf{b}, \mathsf{a}\rangle, \langle\langle \mathsf{b}, \mathsf{b}\rangle, \mathsf{a}\rangle)$ and the state fact indicating that the process on channel c_1 has not yet started are not in mutex, and can be used to trigger the planning rules leading to $\mathsf{att}(\langle \mathsf{b}, \mathsf{a}\rangle, \langle\langle\langle \mathsf{b}, \mathsf{b}\rangle, \langle \mathsf{b}, \mathsf{b}\rangle\rangle, \mathsf{a}\rangle)$. Since the term computed on the Q's side grows at each step, this computation is endless.

Here, it is easy to see that $\mathcal{K}_P$ is not trace included in $\mathcal{K}_Q$, as an attacker can distinguish between b and $\langle \mathsf{b}, \mathsf{b}\rangle$. So, as soon as a message is outputted, the resulting frames are not in static inclusion. Therefore, termination can easily be retrieved by enforcing SAT-Equiv to stop the exploration of the planning graph as soon as an attack is found.

We can turn this example into a more complex one on which the original SAT-Equiv will not terminate even if we decide to stop the exploration of the planning graph as soon as an attack is found. Consider the processes $P_0(c)$, $P_1(c)$ and $Q_1(c)$ given below. We assume that k is name representing a symmetric secret key, whereas $\mathsf{a}, \mathsf{b}, \mathsf{c}$ are public constants.

$$\begin{aligned} P_0(c) &= \mathsf{in}(c, x).\mathsf{out}(c, \mathsf{senc}(x, k)) \\ P_1(c) &= \mathsf{in}(c, \langle \mathsf{senc}(\mathsf{a}, k), \mathsf{senc}(\mathsf{b}, k), \mathsf{senc}(\mathsf{c}, k)\rangle_3).P(c) \\ Q_1(c) &= \mathsf{in}(c, \langle \mathsf{senc}(\mathsf{a}, k), \mathsf{senc}(\mathsf{b}, k), \mathsf{senc}(\mathsf{c}, k)\rangle_3).Q(c) \end{aligned}$$

We consider the configurations $\mathcal{K}'_P = \{P_0(c_0); P_0(c_1); P_1(c_2); P_1(c_3)\}$ and $\mathcal{K}'_Q = \{P_0(c_0); P_0(c_1); Q_1(c_2); Q_1(c_3)\}$ where c_0, c_1, c_2, c_3 are public channel names. Processes P_0 on channels c_0 and c_1 are used as oracles. Roughly, we can get two ciphertexts among the three ciphertexts: $\mathsf{senc}(\mathsf{a}, k)$, $\mathsf{senc}(\mathsf{b}, k)$, and $\mathsf{senc}(\mathsf{c}, k)$. It is however not possible to get the three of them. Noticing this, it is then easy to see that $\mathcal{K}_P$ and $\mathcal{K}_Q$ are trace included.

However, as in the previous example, the planning graph is not precise enough to detect that it is not possible to obtain these three ciphertexts. Once the inputs on channel c_2 and c_3 are executed, we reach a situation similar to the one discussed in the previous example. Each time bad will be added into the planning graph, our SAT encoding will tell us that this state is not truly reachable (but only exists in the over-approximation). Thus, we will continue to explore the planning graph for ever since no attack will be found (the protocols are trace-equivalent).

F Benchmark

In this Appendix, we present the results of our experiments on bounded equivalence checkers. Figure 3 compares the results of SAT-Equiv as in [20] with the current version on three symmetric protocols. Figures 4, 6, 7, as 5, are results on examples of symmetric protocols on which we compared the tools. We extended our benchmark to some assymmetric protocols. The corresponding results can be found on Figures 8 and 9.

Yahalom-Lowe	CSF'17	Current
3	5s	300ms
6	3m	800ms
7	19m	1.9s
10	206m	5s
12	19h	13s
14	TO	23s

Otway-Rees	CSF'17	Current
3	104s	900ms
6	46m	3s
7	50m	5s
10	276m	11s
12	9h40m	16s
14	TO	27s

Simple stateful	CSF'17	Current
1	200ms	20ms
2	1s	80ms
3	2s	150ms
4	6s	200ms
12	155s	800ms
36	85m	3s
60	6h40m	9s

Fig. 3. Comparison of new and old version of Sat-Equiv for Yahalom-Lowe protocol, Otway-Rees protocol and the simple stateful example as in [20].

YP	Spec	Akiss	Deepsec	CSF'17	Sat-Eq	
3	23m	7s	<10ms	50s	400ms	73
6	MO	TO	900ms	165m	5s	122
7			6s	TO	17s	136
10			85m		63s	185
12			TO		143s	214
14					6m	248
21					155m	360
28					7h	472

Fig. 4. Analysis of the Yahalom-Paulson protocol. The second column of Sat-Eq indicates the theoretical bound of the planning graph.

Denning-Sacco	Spec	Akiss	Deepsec	CSF'17	Sat-Eq	
3	12s	80ms	<0.01s	300ms	70ms	42
6	5h	9s	<0.01s	1s	100ms	64
7	MO	75s	<0.01s	2s	200ms	74
10		SO	0.01s	4s	300ms	114
12			0.04s	7s	400ms	134
14			0.2s	11s	500ms	152
21			18s	60s	1.3s	216
28			25m	3m30s	3s	280
35			TO	9m	6s	344
42				23m	10s	408
63				164m	51s	600
84				13h	164s	792
91				23h	4m15s	856
98				TO	6m	920
105					8m	984
126					20m	1176
140					35m	1304
154					56m	1432
168					75m	1560
182					128m	1688
196					3h10m	1816
210					4h20m	1944

Fig. 5. Analysis of the Denning-Sacco protocol. The second column of Sat-Eq indicates the theoretical bound of the planning graph.

WMF	Spec	Akiss	Deepsec	CSF'17	Sat-Eq	
3	6s	40ms	<10ms	100ms	10ms	29
6	58m	1.6s	<10ms	1s	20ms	42
7	TO	5s	<10ms	2s	70ms	47
10		8m30s	60ms	7s	100ms	60
12		SO	440ms	40s	200ms	69
14			5s	118s	350ms	78
21			21m	77s	200ms	106
28			TO	4m45s	400ms	137
35				24m	600ms	168
42				36m	1s	199
56				163m	2.4s	261
63				5h	4s	292
70				9h	5s	323
77				18h	7s	354
84				TO	9s	385
112					24s	509
140					55s	633
154					93s	695
168					140s	757
182					3m30s	819
196					4m30s	881
210					6m	943

Fig. 6. Analysis of the Wide-Mouth-Frog protocol. The second column of Sat-Eq indicates the theoretical bound of the planning graph.

NSS	Spec	Akiss	Deepsec	CSF'17	Sat-Eq	
3	1m	4s	<10ms	2s	80ms	50
6	MO	TO	10ms	54s	400ms	88
7			50ms	153s	1.2s	98
10			600ms	8m	3s	132
12			7s	22m	4s	155
14			120s	77m	11s	176
21			TO	TO	51s	346
28					178s	452
47 ref.					48m	527
47					129m	759
94 ref.					20h30m	1024
94					MO	1489

Fig. 7. Analysis of the Needham-Schroeder protocol. The second column of Sat-Eq indicates the theoretical bound of the planning graph. For the examples with 47 and 94 roles, we also consider refined scenarios where some roles are cut before the end, as explained in Subsection 6.2.

PA	Spec	Akiss	Deepsec	Sat-Eq	
2	3s	100ms	<10ms	10ms	27
4	14m	4s	10ms	30ms	43
6	MO	10m	20ms	60ms	59
8		TO	20ms	90ms	75
10			80ms	100ms	91
20			3s	300ms	171
40			3h20m	800ms	332
46			TO	1s	380
50				1.1s	412
54				1.2s	444
60				1.7s	492
80				3s	652
120				6s	972
160				12s	1292
200				20s	1612
400				98s	3212

AA	Spec	Akiss	Deepsec	Sat-Eq	
2	3s	100ms	<10ms	10ms	37
4	15m	3s	<10ms	40ms	61
6	MO	5m	20ms	70ms	85
8		SO	20ms	70ms	109
10			60ms	100ms	133
20			2s	300ms	253
40			2h30m	800ms	493
46			23h50m	900ms	565
50			TO	1.1s	613
54				1.2s	661
60				1.7s	733
80				3s	973
120				6s	1453
160				11s	1933
200				18s	2413
400				78s	4813

Fig. 8. Analysis of the Passive-Authentication protocol (left) and Active Authentication (right). The second column of Sat-Eq indicates the theoretical bound of the planning graph. For Passive Authentication, the verified property is anonymity.

NSL	Spec	Akiss	Deepsec	Sat-Eq	
2	11s	40ms	10ms	20ms	27
4	MO	2s	10ms	40ms	43
6		SO	120ms	200ms	59
8			7s	500ms	75
12			58m	0.9s	107
16			TO	3s	139
32				35s	267
48				162s	395
64				11m	523

DS sig.	Spec	Akiss	Deepsec	Sat-Eq	
2	1.5s	60ms	<10ms	0.01s	39
4	4m	1.5s	<10ms	0.02s	55
8	MO	SO	240ms	0.2s	87
16			4h50m	0.9s	151
18			TO	1.1s	167
24				3s	215
32				8s	279
48				66s	407
64				100s	535

Fig. 9. Analysis of Needham-Schroeder-Low protocol (left) and Denning-Sacco protocol with signature from [6] (right). The second column of Sat-Eq indicates the theoretical bound of the planning graph.